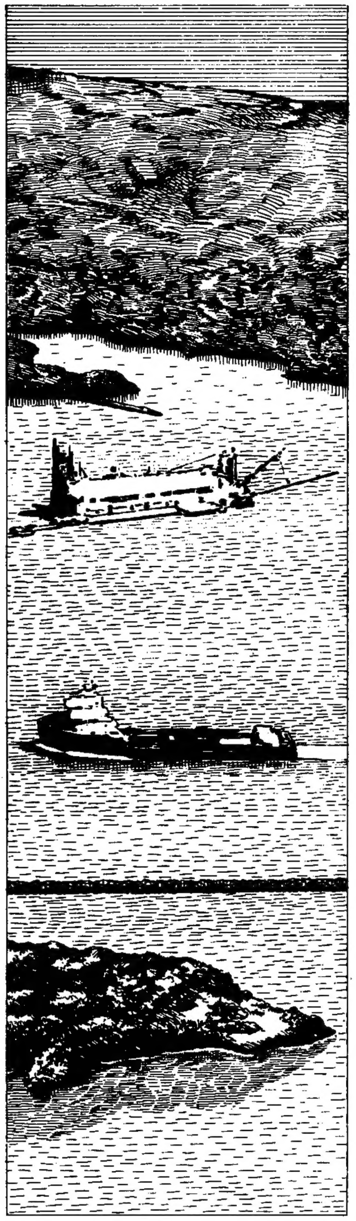
US Army Corps of Engineers



DREDGING RESEARCH PROGRAM

TECHNICAL REPORT DRP-96-1

SILENT INSPECTOR SYSTEM TECHNICAL MANUAL

by

Jeffrey M. Cox

Evans-Hamilton, Inc.
731 Northlake Way, Suite 201
Seattle, Washington 98103

Paul Maresca

AdaSoft, Inc.
8750-9 Cherry Lane, Laurel, Maryland 20707

James Rosati III

DEPARTMENT OF THE ARMY
Waterways Experiment Station, Corps of Engineers
3909 Halls Ferry Road, Vicksburg, Mississippi 39180-6199

February 1996

Final Report

Approved For Public Release; Distribution Is Unlimited

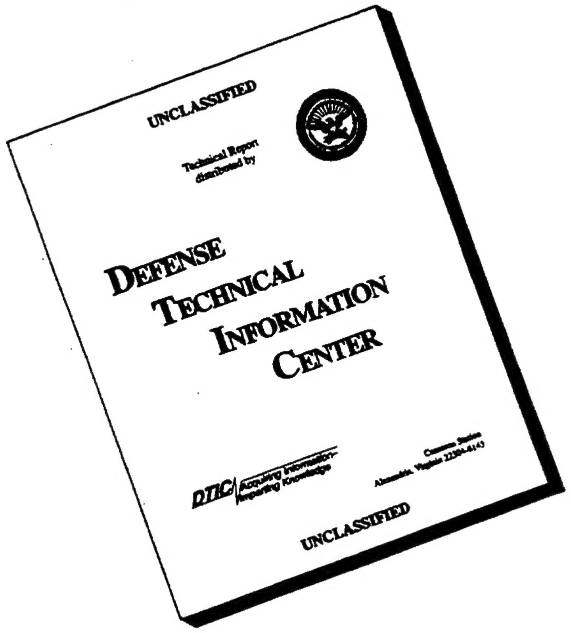
19960307 024

Prepared for DEPARTMENT OF THE ARMY
U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under Work Unit 32482

DTIC QUALITY INSPECTED 1

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

The Dredging Research Program (DRP) is a seven-year program of the U.S. Army Corps of Engineers. DRP research is managed in these five technical areas:

Area 1 - Analysis of Dredged Material Placed in Open Water
Area 2 - Material Properties Related to Navigation and Dredging
Area 3 - Dredged Plant Equipment and Systems Processes
Area 4 - Vessel Positioning, Survey Controls, and Dredge Monitoring Systems
Area 5 - Management of Dredging Projects

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products.

PRINTED ON RECYCLED PAPER

US Army Corps of Engineers
Waterways Experiment Station

Dredging Research Program Report Summary

Silent Inspector System Technical Manual (TR DRP-96-1)

ISSUE: Inspection of dredging operations can be greatly assisted by automatic logging of dredging activities and creation of dredging reports. Development of a system to assist inspectors in this capacity was the goal of this project.

RESEARCH: A Dredge Operations Silent Inspector System (DOSIS), developed to automatically log data from instruments generally maintained aboard hopper dredges, compute the dredging activities occurring and the quantity of material retained, and provide summaries of this information in both reports and graphical displays, has been developed and is undergoing testing and improvements. While the system was designed for use aboard hopper dredges, its basic concept will be applied to other dredge types in the future.

SUMMARY: This report describes how the Silent Inspector system for hopper dredges operates internally assuming the system has already been installed by a systems engineer. This report is intended to be used by systems engineers. A companion *Silent Inspector User's Manual*[1] describes how to operate the installed system.

AVAILABILITY OF REPORT: The report is available through the Interlibrary Loan Service from the U.S. Army Engineer Waterways Experiment Station (WES) Library, telephone number (601) 634-2355. National Technical Information Service (NTIS) report numbers may be requested from the WES library.

To purchase a copy of the report, call NTIS at (703) 487-4780.

---

[1] J. M. Cox, P. Maresca, and A. Jarvela. (1995). "Silent Inspector User's Manual," Instruction Report DRP-95-2, U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS.

About the Authors: Mr. Jeffrey Cox, Evans-Hamilton, Inc., oversees the team developing the Silent Inspector system, working under the direction of Mr. James Rosati, the DRP Principal Investigator for the Silent Inspector Work Unit. Mr. Cox wrote the majority of the report. Mr. Paul Maresca, AdaSoft Inc., leads the system software development, and herein provided writing, technical guidance, and editing concerning the operating characteristics of the Silent Inspector system. This work is directed by Mr. James Rosati of the Prototype Measurements and Analysis Branch, Coastal Engineering Research Center, located at the U.S. Army Engineer Waterways Experiment Station. For further information about the Silent Inspector system, or the DRP, contact either Mr. Rosati at (601)634-2202, or Mr. E. Clark McNair, Jr., Manager of DRP, at (601)634-2070.

Dredging Research Program

Technical Report DRP-96-1
February 1996

Silent Inspector System Technical Manual

by Jeffrey M. Cox

Evans-Hamilton, Inc.
731 Northlake Way, Suite 201
Seattle, WA 98103

Paul Maresca

AdaSoft, Inc.
8750-9 Cherry Lane
Laurel, MD 20707

James Rosati III

U.S. Army Corps of Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

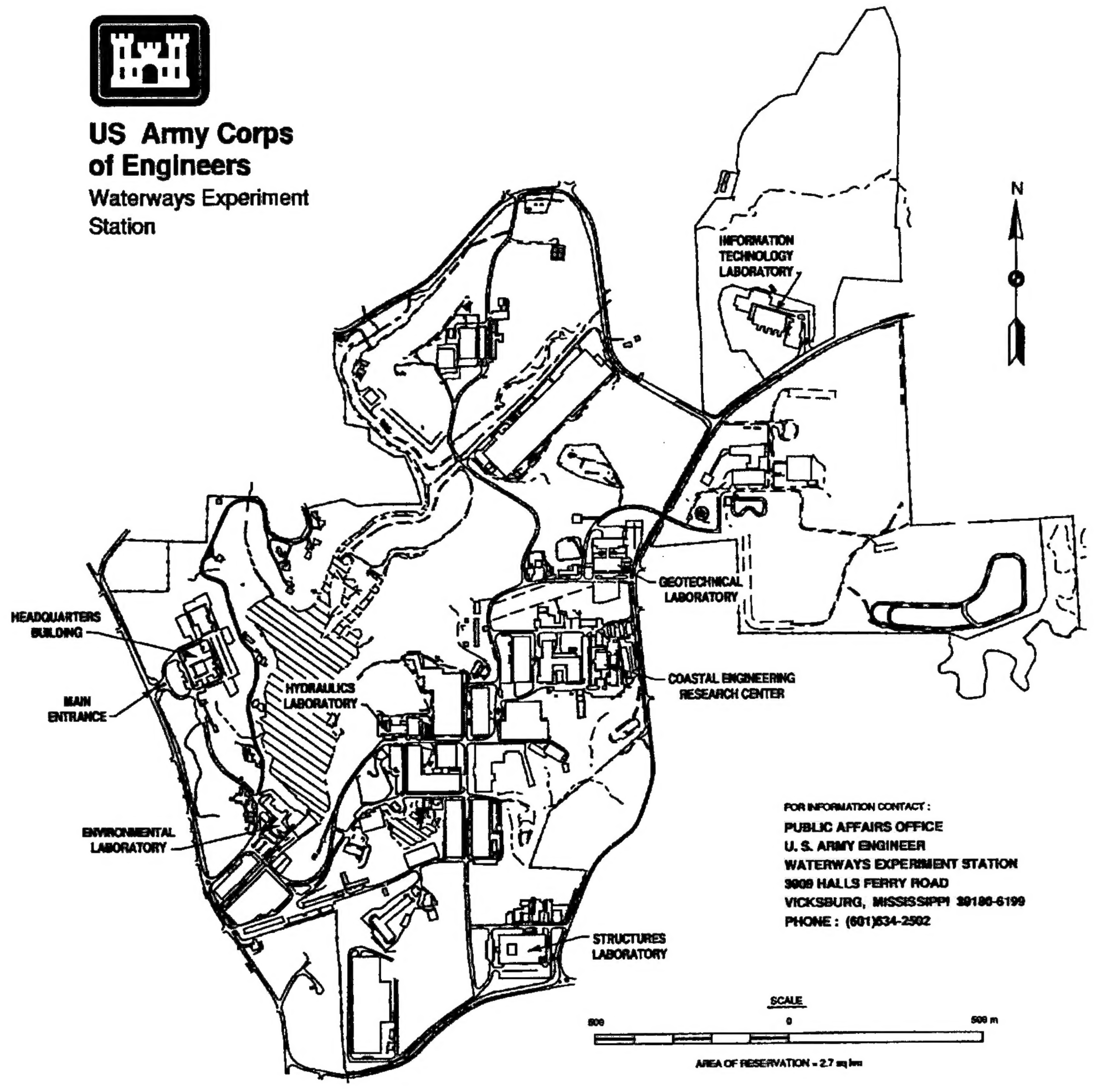
Final report

Approved for public release; distribution is unlimited

Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under Work Unit 32482

US Army Corps of Engineers
Waterways Experiment Station



FOR INFORMATION CONTACT:
PUBLIC AFFAIRS OFFICE
U. S. ARMY ENGINEER
WATERWAYS EXPERIMENT STATION
3909 HALLS FERRY ROAD
VICKSBURG, MISSISSIPPI 39180-6199
PHONE: (601)634-2502

Waterways Experiment Station Cataloging-in-Publication Data

Cox, Jeffrey M.
Silent Inspector System technical manual / by Jeffrey M. Cox, Paul Maresca, James Rosati III ; prepared for U.S. Army Corps of Engineers.
126 p. : ill. ; 28 cm. — (Technical report ; DRP-96-1)
1. Dredging — Handbooks, manuals, etc. 2. Dredges — Computer programs —Handbooks, manuals, etc. 3. Silent Inspector System (Computer program) — Handbooks, manuals, etc. I. Maresca, Paul. II. Rosati, James. III. United States. Army. Corps of Engineers. IV. U.S. Army Engineer Waterways Experiment Station. V. Dredging Research Program (U.S.) VI. Title. VII. Series: Technical report (U.S. Army Engineer Waterways Experiment Station) ; DRP-96-1.
TA7 W34 no.DRP-96-1

Contents

List of Figures

List of Tables

Preface

This technical manual was prepared for the U.S. Army Engineer Waterways Experiment Station (WES) under Dredging Research Program (DRP) Technical Area 4, Work Unit No. 32482, "Silent Inspector." The DRP is sponsored by Headquarters, U.S. Army Corps of Engineers (HQUSACE). HQUSACE Technical Monitor for Technical Area 4 was Mr. M. K. Miles.

This technical manual was written by Mr. Jeffrey Cox, Evans-Hamilton, Inc., Seattle, WA; Mr. Paul Maresca, AdaSoft, Inc., Laurel, MD; and Mr. James Rosati III, WES. Technical oversight was provided by Mr. James Rosati, Principal Investigator, Prototype Measurements and Analysis Branch, Coastal Engineering Research Center (CERC), WES. Mr. E. Clark McNair, Jr., and Dr. Lyndell Z. Hales were Manager and Assistant Manager, respectively, of the DRP. Dr. James R. Houston and Mr. Charles C. Calhoun, Jr., were Director and Assistant Director, respectively, of CERC, which conducted the DRP.

At the time of publication of this report, Dr. Robert W. Whalin was Director of WES. COL Bruce K. Howard, EN, was Commander.

For further information concerning this report, contact Mr. James Rosati, (601) 634-2022, or Mr. E. Clark McNair, Jr., (601) 634-2070.

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products.

Summary

This report describes the Silent Inspector system developed for monitoring hopper dredge operations. The system collects and records measurements from shipboard sensors, calculates the dredging activities being performed and the weight of the material recovered, and displays this information through standard reports and graphical data displays. Recorded data are also automatically backed up and later archived to allow transfer of the data to other locations. The Silent Inspector data can provide a permanent record of the dredging activity.

The system operates on personal computers using the Unix operating system to allow multitasking operations, and consists of three primary components: a dredge-specific software (SDD) component, a ship-based component (SHIP), and a shore-based component (SHORE). The DSS collects sensor data, checks these data against acceptable ranges, computes the status of the dredging pumps (on/off) and other equipment, attaches the name of the project, dredge, and contract number to the sensor data, and inserts these incoming data into the system's central database. SHIP maintains the system's central database, accepting data in real time from a DSS. SHIP then reviews those data, computes the present dredging activity being performed and the amount of material recovered, and produces reports (trip, daily, job) and graphical displays of the data. Additional information concerning the dredging project, the dredges used, and location of the dredging and disposal areas can also be inserted into the system database via the SHIP component. SHORE differs from the SHIP component in that it does not receive data in real time from a DSS unit. Its purpose is to provide in-office data review, display, and post-field processing.

The system is built upon the Sybase relational database and utilizes client-server architecture to interface with programs and computers accessing the database. Once turned on, the system operates nearly autonomously in its collection, processing, reporting, and archiving of data. Many additional operations, such as specific data displays, are user initiated and controlled. Nearly all user controlled activities are operated using graphical screens for ease of use and minimal training time. System security is maintained through user ID's and password protection.

Described herein are the system's database structure, data computations, data flow, system architecture and interfaces, and software structure. This manual is designed to assist system engineers in the upkeep and modification of the system and its components.

1 Introduction

Concept of the Silent Inspector System

The Dredging Operations Silent Inspector System (DOSIS), referred to as the Silent Inspector, was created to be a tool to assist Corps dredging inspectors in monitoring the performance of contracted hopper dredging operations on a 24-hr basis. The Silent Inspector provides automated collection, analysis, display, and reporting of information about a dredge's activities which allow inspectors and contract administrators to better assess contractor performance and adherence to contract terms, both when dredge inspectors are aboard the dredge and especially when they are not aboard. The present system, described in this manual, is designed for hopper dredges only.

The system uses state-of-the-art computer hardware and software to simultaneously measure several parameters of the dredge's activities, calculate additional information, such as the activity the dredge is performing and the quantity of material it has retained, display this information, automatically produce Corps dredging reports, and provide a permanent and transferable copy of this information. The system strives to produce a factual record of the dredge's activities that is sufficient for dredging inspectors to accurately assess contract performance.

In particular, the overall functions the system is designed to perform are:

a. Record data on several aspects of dredging operations.

b. Properly label the data with the project name, contract ID, dredge name, trip (load) number, and dredging and disposal locations.

c. Compute the type of dredging activity occurring aboard the dredge at any time and the amount of material recovered in terms of tons dry measure (TDM).

d. Automatically provide summaries of trip (load), daily, and job-to-date dredging activities on a basis similar to reports in use today.

e. Graphically display recorded data.

f. Automatically back up recorded data daily and weekly.

g. Archive the data for transfer to other Silent Inspector systems and for permanent record keeping.

While the direct purpose of the system is to allow inspectors and contract administrators to assess contract performance, use of the Silent Inspector system provides the following additional benefits to the Corps of Engineers:

a. Measures, records, and reports dredging information in a standardized format for similar types of dredges to allow the data to be transferred easily between all Corps of Engineers Districts and offices.

b. Provides a means to electronically archive the data collected and transfer it to users working on other hardware and software mediums.

c. Provides a graphical display of collected data by time or location.

d. Provides a Corps-generated independent database of dredging activities for use in settling claims brought by contractors against the Corps.

e. Provides flexibility and extendibility to meet changing system requirements, and to meet each District's unique system requirements.

f. Allows Corps users to add individual features without destroying the integrity or operation of the basic system.

g. Provides the Corps a more accurate assessment of dredging activities, downtime, and production rates for more accurate estimating of future project costs and budgets.

The system has been designed to be easy to use via graphical user screens, is highly automated so that it functions reliably, and noncorruptible, so that access to the system is limited to approved users, and the recorded data cannot be altered or manipulated.

About This Manual

To assist in the development of the Silent Inspector system operating requirements and standards, a training and reference version of the system was developed. This system is a specific implementation of the Silent Inspector concept. This manual provides technical information on how the training and reference Silent Inspector system for hopper dredges is constructed and functions. This manual is specifically targeted to system support personnel and system engineers and programmers. This manual does not describe how to operate the system; for that information, the reader is referred to the *Silent Inspector User's Manual.*[1]

This manual is organized into the following chapters:

a. *System Overview*: Chapter 2 discusses the system's overall capabilities, its major components, its software modules and functions, and the software standards to which it adheres.

b. *Dredge-Specific System*: Chapter 3 describes the function and actions of the dredge-specific software (DSS) component of the system, which gathers sensor measurements and inputs these data into the system database.

c. *System Database*: Chapter 4 describes the system's central database structure, tables, data flow, and controlling programs.

d. *System Computations*: Chapter 5 lists the computations and provides the equations for all calculations of new information which occur within the system.

e. *System Interfaces*: Chapter 6 describes the interface requirements and formats used by, and required for interaction with, the system.

f. *SHIP Software Modules*: Chapter 7 describes how various software modules controlled by the user function, once initiated.

g. *Stored Procedures*: Appendix A is a listing of the stored procedures used with the Silent Inspector software.

h. *Backup and Archive Script*: Appendix B is a listing of the backup and archive script for the Silent Inspector software.

i. *Configuration Management Library Structure*: Appendix C is a listing of the configuration management library structure for the Silent Inspector software.

[1] Cox, J. M., Maresca, P., and Jarvela, A. (1995). "Silent Inspector User's Manual," Instruction Report DRP-95-2, U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS.

2 System Overview

This chapter provides an overview of how the hopper dredge Silent Inspector system is structured, how data move through the system, where data are stored, and the functions of the various software modules.

System Operational Components

The Training and Reference Silent Inspector system is composed of three operational units, called components. These components are named:

a. Dredge-specific software component (DSS).

b. Ship component (SHIP).

c. Shore component (SHORE).

The central database and primary data processing actions are contained within the SHIP component. This component receives data in real time from the DSS component, uses those data to compute the dredge's current activity (dredge state) and other information, and packages those data into standard reports and displays. The DSS component accesses sensor data on dredges, checks that the data are within an acceptable range, and repackages the data for insertion into the SHIP database. The SHORE component accepts data which have been acquired and processed by a SHIP database, and allows users to review and display previously recorded data, and regenerate dredging reports over user-specified time periods. A more detailed overview of each component is provided below.

Dredge-specific software (DSS) component

The Silent Inspector system's central database for receiving and storing real-time data aboard a dredge resides within the SHIP component. This database is designed to accept specific types of data about hopper dredge operations. In addition, the SHIP component is designed to be identical in all Silent Inspector hopper dredge systems, so that the system is consistent between all dredges. As each hopper dredge contains a different suite of measurement sensors, sensor locations, calibrations, data formats, and types of data available, these unique packages of sensor data must be converted into a standard package of data, called data records, and sent to the SHIP component in identical format and fashion. This is the role of the DSS component. The specific functions of the DSS are to:

a. Acquire dredge sensor measurement data every 10 sec.

b. Determine if the sensor data are within acceptable ranges.

c. Compute additional information needed by the SHIP database.

d. Repackage measured and computed data into a standard format (data record).

e. Attach the contract ID, project name, and dredge name provided by the user to the data record.

f. Insert the data into the SHIP database.

While the output of each DSS is identical, the inputs to a DSS will vary from dredge to dredge; therefore, no two DSS units are expected to operate internally in exactly the same way. This manual describes the standard portions of the DSS component, such as the DSS-to-SHIP interface, as well as the unique portions of the training and reference DSS, which was created to interact with the Corps of Engineers dredge *Essayons* for purposes of system development and testing.

SHIP component

The SHIP component is the heart of the Silent Inspector system. Like the DSS, it resides onboard the dredge; however, all SHIP components are identical. Required measured data are received from the DSS and are inserted into the system's central database. The SHIP system then also performs the following:

a. Displays the incoming data in real time.

b. Automatically computes and records the current dredging operation, called the dredge state, as well as the amount of material retained (TDM).

c. Automatically creates standard trip and daily reports based on the recorded data, and creates job-to-date reports when initiated by the user.

d. Graphically displays any measurement data contained within the database that are requested by the user.

e. Automatically backs up and archives data.

f. Permits the user to input information concerning the dredging project, dredge, its crew, dredging and disposal areas, and local marine landmarks.

The above capabilities are either initiated or controlled by several software modules. These modules generally allow the user to input specific information into the database, display or review data within the database, or control the automatic operation of the database. The primary software modules contained within the SHIP component, and their functions, are:

Software Modules	Purpose/Function
RT Kernel	Computes dredge state, TDM, computes and prints standard reports, controls flow of data through database
Setup	User entry of project and dredge information
Monitoring	Displays incoming data from DSS
Standard Reports	Calculates and produces trip and daily reports
Downtime	Display of downtime events, and user input of downtime cause and comments
Plotting	Graphical display of recorded data over time period selected by user
Backup	Automated daily and weekly backup of recorded data, and user initiated restoration of database using backup files
Archive	Automated archiving of recorded data, and user initiated transfer of archived files

SHORE component

When data, which are recorded on a dredge by the SHIP component, are to be analyzed at a location other than on the dredge, the data are transferred to a SHORE component. The SHORE component may reside at any location, such as the Corps of Engineers District offices. Data analysis may include more detailed review of the data, and inter-comparison of data obtained from multiple SHIP units.

The SHORE component functions quite similar to SHIP, with the following exceptions:

- It operates on shore-based computers.
- It does not collect data in real time from a DSS component.
- It is able to input and review data sets from multiple dredging projects.

Software modules present within the SHIP component (listed above) but not present or active within the SHORE component are:

Software Modules	**Purpose/Function**
RT Kernel	Computes dredge state, TDM, computes and prints standard reports, controls flow of data through database
Monitoring	Displays incoming data from DSS
Archive	User-initiated loading or downloading of archived files

The SHORE user interface screens are identical to the SHIP user interface screens for those software functions contained in both components. Multiple SHIP data sets can be input into SHORE. Each remains unique based on the project name, contract ID, and dredge name assigned to the data.

Hardware and Software Requirements

The Training and Reference Silent Inspector system for hopper dredges operates on the Unix operating system, and 486 type personal computers (PC's). The system software has been designed to require minimal or no user knowledge of the Unix operating system or commercial software packages utilized within the system. User interactions with the system are via a graphical user interface that allows easy setup, operation, and data review. Currently the system utilizes the hardware and software listed below.

System hardware

The Silent Inspector hopper dredge system is designed to be used on PC's, because they are commonly used throughout Corps of Engineers Districts. Communication between components, if the components reside on separate computers, is via Ethernet when the component computers are cabled directly together. The following computer hardware is necessary for each component to ensure proper operation:

Hardware	*DSS*	*SHIP*	*SHORE*
486 PC with minimum of 33-MHz clock, VGA graphics, standard monitor, 500-MB hard drive, 32-MB RAM	x	x	x
Ethernet link	x	x	
Postscript laser printer		x	x
Streaming tape drive and tapes		x	x
Uninterruptable power supply (UPS)		x	x

System software

The Silent Inspector system was developed using several commercial software packages. These packages provide the graphical user screens and database engine within the system. Consequently, each component of the system needs to access these commercial packages to operate properly. The software packages listed below must be purchased and installed on the DSS, SHIP, and SHORE computers for the system to work.

Software
SCO Unix with open desktop
SL-GMS
Sybase relational database management system (DBMS) (2-8 users)

System-specific software is used to tie specific user actions on the graphical user interface screens to system actions and to control the flow of data through the database and display screens. In the Training and Reference Silent Inspector system, this system-specific software is written primarily in the ADA software language. Some limited commands also are programmed within Unix script. Database stored procedures utilize Sybase commands and script.

System Development Standards

The following standards were used in the development of the Training and Reference Silent Inspector System. All implementations of the Silent Inspector system for hopper dredges should adhere to these standards. Some of these standards are recognized industry standards, and some were developed as part of the Silent Inspector development program to ensure common capabilities between all Silent Inspector systems, regardless of their specific implementation conditions.

Industry standards

Operating System:
IEEE Posix 1003.1 and FIPS 151-1

Database Management System:
SQL (FIPS 127, ANS X3.135-1986)
ISO-RDA (Remote Database Access)

Graphics Data Interchange:
CGM (FIPS 128)
IGES
PDES (NBSIR 88-3813)
Postscript

Network Services:

OSI GOSIP (ANSI X3T5 and FIPS 146)
NFS (IEEE Posix 1003.8)

User Interface:

X Window System - X11.6 (FIPS 158)
OSF/Motif
Microsoft's Presentation Manager

Connectivity:

TCP/IP
Ethernet (IEEE 802.3)

Applications Portability Profile:

FIPS 151 (with conformance to extended Posix, SQL, OSI, NFS, X-Windows, C and other languages)

Program Languages:

C	(ANSI X3J11 and X3.159-1989)
FORTRAN	(FIPS 069-1 and ANSI X3J3, ANS X3.9-1978)
ADA	(FIPS 119, ISO 8652, ANSI MIL 18115A)
PASCAL	(FIPS 109, ANSI S3I9, ISO 7185-1983 Level 1)

Silent Inspector specific standards

Interfaces:

SHIP database input/DSS output

Reports:

Trip
Daily
Job-to-date

Computations:

TDM
Dredge state

The specific requirements of each of the system-specific standards are described in detail within later chapters in this manual.

3 Dredge-Specific System

The dredge-specific system (DSS) component of the Silent Inspector system is designed to receive a variety of measurements from sensors aboard hopper dredges, to formulate from these sensor measurements a common (identical) package of data required by the SHIP component of the system, and to insert this package of data into the SHIP database at regular (10-sec) intervals. To perform this function, the DSS computes some data values which are not directly measured. The DSS also compares each measured or calculated data value to an acceptable range for that data value and attaches a data status flag to the data within the SHIP database.

Because no two hopper dredges are expected to have identical sensor information available or to necessarily input the available measurements into the DSS in identical fashion (RS232, RS422, digitized voltages, etc.), each DSS is expected to differ in how it handles accepting sensor data, internal data computations, acceptable data ranges, necessary operator inputs, and data display. All DSS components are expected to be identical in terms of some operator inputs, data status checking procedures, content and format of the data sent to the SHIP database, and how it inserts the data into the SHIP database (DSS-to-SHIP interface).

The specific tasks the DSS performs are listed below:

a. Handling of operator inputs.

b. Reading of required sensor data.

c. Conversion of the sensor data from its input format to the format required for insertion into the SHIP database.

d. Computation, based on the sensor data, of additional data required to be inserted into the SHIP database.

e. Assignment of status values to applicable sensor and computed data values.

f. Display of the sensor and computed measurements and status values.

g. Insertion of the sensor and computed data and associated status values into the SHIP database.

How each of these tasks is performed within the training and reference DSS is described in detail in the following sections.

Operator Inputs

All DSS units require the user to input three pieces of information about a project: the contract ID, project name, and dredge name. Each item may be up to 32 characters in length. This information is entered on the DSS Entry Panel screen (see *User's Manual*, Figure 4.4).[1] The contract ID, project name, and dredge name are inserted into the SHIP database with every sensor input record to identify the source of the data.

The training and reference DSS also allows the user to select or control two other items for use in testing the system. These are: (a) the source of incoming sensor data (either real-time from a dredge, or from a pre-recorded computer file) and (b) the time associated with the incoming data when they are sent on to the SHIP database.

To select a data file as the source of incoming data, the user must enter computer drive, directory, subdirectory, and file name within the data source box on the DSS Entry Screen. The DSS will then access and read the data from that file. If no entry is made within the data source box, the DSS assumes the sensor data are entering the DSS through an RS232 port in real-time.

If a file is being used as the input source, the sensor data read from the file are processed and inserted into the database. Each set of sensor data has an original time tag associated with it. For testing purposes, it is sometimes useful to set a data rate different from that at which the sensor data were originally collected. This can be done by the operator by clicking on the user-defined timer input button and specifying a time interval between measurements. A slider bar is displayed that permits the operator to set the data interval in seconds.

When a user-defined time interval is set, the original time tag associated with the first set of measurements is used to establish the date/time of the first measurement set, and then all other date/time values associated with the remaining measurements in the file are computed by the DSS using the user-selected time interval. The time tag for each successive set of measurements is determined by adding the user-defined time interval to the time tag computed for the immediately preceding measurement set. The original date/time values

[1] *Ibid.*

are not altered within the pre-recorded data file; however, the newly computed date/time values are inserted along with the data into the SHIP database.

Reading Sensor Data

The training and reference DSS was first tested on the Corps of Engineers dredge *Essayons*; therefore, the sensor data input format is specifically tuned to this dredge and the output of a data logging computer presently operating aboard the dredge. The following sensor data are currently accepted into the training and reference DSS. This same suite of sensor data should be input into all DSS units; however, it is understood that on some hopper dredges this suite of sensor measurements may not be currently available.

Data Element	**Units**
Date	Local time
Time	Seconds, from midnight local time
RMS error (of ship's position)	feet
X location	feet
Y location	feet
Forward draft (of ship)	feet
Aft draft (of ship)	feet
Tide elevation	feet (relative to MLLW)
Port drag arm velocity	feet/sec
Port drag arm density	g/l
Starboard drag arm velocity	feet/sec
Starboard drag arm density	g/l
Port gimbal depth	feet
Starboard gimbal depth	feet
Port draghead depth	feet
Starboard draghead depth	feet
Heading (of ship from Gyro)	degrees True
Course (of ship)	degrees True
Water depth (below ship)	feet
Speed (of ship over ground)	knots
Ship weight-hopper doors open	long tons
Ship weight-hopper doors closed	long tons
Ullage (meter #1)	feet
Ullage (meter #2)	feet
Ullage (meter #3)	feet
Ullage (meter #4)	feet
Hopper door status	open/closed

The DSS is capable of accepting the above sensor data as a continuous input stream through an RS232 input port or from a pre-recorded ASCII text file. The latter mode is normally used for testing the training and reference Silent Inspector system, and this capability is not expected to be included in standard systems.

Data received through the RS232 port are in binary format and are normally received from another computer. When real-time sensor data are received via the RS232 serial port, the DSS divides the data stream up into records by treating some number of contiguous bytes as a record. In the training and reference DSS, this quantity is 127 bytes, with a record being defined as a set of sensor measurements all associated with a single time tag. The format and position of each sensor measurement value within each record are described in Chapter 6 ("System Interfaces") of this manual. The input data stream is first broken into individual records, which are collected in a buffer. Next, the sensor values within each record are extracted and placed within a database structure, defined within the DSS system code. The contract ID number, project name, and dredge name for this project, previously entered into the DSS by the operator, are then attached to each data record stored in the DSS database structure. Next, the sensor values in a record are used to compute some additional data values, and all values are cross-checked against a limits table to assign a data status flag to several of the data values. These computations are listed below. The newly computed values and status flags are then attached to the data record in the DSS database structure. Each data record is then inserted into the SHIP database.

If the input source is a data file, each record is read as a string of ASCII text. The fields that represent the sensor measurements are extracted from the string and placed within the DSS database structure. Again, the contract ID number, project name, and dredge name for this project, previously entered into the DSS by the operator, are attached to each data record; and the sensor values in the record are used to compute additional data values and data status flags. The newly computed values and status flags are then attached to the data record in the DSS database structure. In addition, the time previously attached to the data record is disregarded, except for the time of the first data record read. All subsequent times attached to the incoming records are based upon the time of the first data record and the time interval between data records selected by the operator on the DSS control panel screen. This feature allows for more rapid testing of the training and reference Silent Inspector system. After a new time is attached to the record, the data record is then inserted into the SHIP database.

The DSS accepts data records from either the RS232 port or a pre-recorded data file as fast as the data logging computer can send them and the DSS can process these data and send them on to the SHIP database. The maximum input speed for the DSS is faster than one data record per second. In the case of the *Essayons*, the data logging computer sends the DSS a new data record every 10 sec. Currently there is no provision for error checking or synchronization, should data be lost in transmission.

Computation of Additional Data Values

The SHIP database also requires as input some data types that are not provided by sensors aboard the *Essayons*. Likewise, it is anticipated that sensors aboard other hopper dredges may not provide the full suite of measurements necessary for insertion into the SHIP database. Where possible, the DSS also must compute the additional data needed by the SHIP database based on the available sensor data. The following computed data are generated within the training and reference DSS:

Computed Data Type	Computed From
Average hopper level forward	Two forward ullage sensor readings
Average hopper level aft	Two aft ullage sensor readings
Average hopper level	Average forward and aft hopper level values
Hopper volume	Average hopper ullage and hopper volume curves/equations
Port/Stbd pumps on/off	Port and starboard draghead velocities
Port/Stbd material recovery true/false	Port and starboard draghead densities
Pump out on/off	Pump-out velocity (not provided on *Essayons*)

The equations for each of these computations are provided in Chapter 5 of this manual.

Assigning Status Values

The DSS provides the first level of data QA/QC for the Silent Inspector system. This process is performed by comparing each sensor or computed data value to an acceptable range for that data value. The status values that can be assigned are:

Status	Abbr.	Meaning
ACCEPTABLE	OK	Sensor measurement is within acceptable limits
OUT_OF_RANGE_LOW	LO	Sensor measurement is out-of-range low
OUT_OF_RANGE_HIGH	HI	Sensor measurement is out-of-range high
MISSING	NO	Sensor measurement is missing

Minimum and maximum acceptable values for each type of data are read from a file named ***RANGE.DAT*** stored within the DSS. This file is an ASCII text file that may be edited to adjust the values for each type of data. The user is not intended to have access to this table. Values in this table must be entered or changed by a systems engineer. The current ranges established for the training and reference DSS for operation aboard the *Essayons* are:

Sensor Measurement	Lower Bound	Upper Bound
RMS_ERROR	0.0	32.9
X_COORDINATE	0.0	9_999_998.0
Y_COORDINATE	0.0	9_999_998.0
FORWARD_DRAFT	10.0	35.0
AFT_DRAFT	15.0	37.0
TIDE	-5.0	35.0
PORT_DRAGARM_VELOCITY	0.0	50.0
PORT_DRAGARM_DENSITY	1.0	3.0
STBD_DRAGARM_VELOCITY	0.0	50.0
STBD_DRAGARM_DENSITY	1.0	3.0
PORT_GIMBAL_DEPTH	0.0	90.0
STBD_GIMBAL_DEPTH	0.0	90.0
PORT_DRAGHEAD_DEPTH	-20.0	90.0
STBD_DRAGHEAD_DEPTH	-20.0	90.0
HEADING	0.0	359.9
COURSE	0.0	359.9
GROUND_SPEED	0.0	16.0

WATER_DEPTH	0.0	90.0
HOPPER_ULLAGE	0.0	50.0

If a measurement value exceeds its upper bound, its corresponding status value is set to OUT_OF_RANGE_HIGH (HI). Similarly, if it is less than its lower bound, its status value is set to OUT_OF_RANGE_LOW (LO). If a measurement is missing, the status MISSING (NO) is assigned to it. Otherwise, the measurement is deemed acceptable, and its status value is set to ACCEPTABLE (OK). These status values are stored within the DSS in the DSS database structure; however, they are inserted into the SHIP as integer values from 0 to 3 as shown below.

Status	**Database Value**
ACCEPTABLE	0
OUT_OF_RANGE_LOW	1
OUT_OF_RANGE_HIGH	2
MISSING	3

The data status values are used to color code the display of the data values within both the DSS and the SHIP components. In addition, the status values are also used in determining which data are acceptable to use in some computations within the SHIP component.

Data Display

The Training and Reference DSS displays on its control panel screen the sensor data being inserted into the SHIP database during processing. Each sensor value is displayed with a background color that indicates its associated status value. These are the data which are being inserted into the SHIP database. The "Remaining" box shows the time remaining until the next data set, which occurs at 10-sec intervals.

If the DSS is receiving data, but the SHIP component is not operational and/or otherwise unprepared to receive data from the DSS, the DSS will attempt to open the SHIP database and insert data records, but will be unable to do so. If the DSS continues to try and open the SHIP database but is unsuccessful for approximately 1 min, the DSS program will then terminate itself, in which case it will need to be restarted after the SHIP database comes on-line.

Data Manipulation for System Testing

The training and reference DSS has a unique feature which will not be present on any standard operational DSS units. The purpose of this feature is to allow for testing of the DOSIS system during its development so as to prevent breakdown or corruption of the system by unacceptable or missing data.

This feature allows false values or no data to be inserted into the SHIP database for each data value displayed on the DSS data monitoring screen.

Beside each data value displayed on the DSS screen is a set of four buttons marked OK, LO, HI, and NO. By clicking on any one of these buttons, the operator can change the status value associated with the corresponding sensor measurement. Clicking on a status button performs the following: (a) it changes the actual data value that is inserted into the SHIP database, as well as the status value associated with it; and (b) it changes the background color used to display the measurement value on the DSS screen to match the value's status. If the operator changes the status of a measurement to LO, the measurement value that is inserted into the SHIP database is set to the minimum value of its acceptable range, obtained from the ***RANGE.DAT*** table, minus 0.1. If the operator changes the status of a measurement to HI, the measurement value that is inserted into the database is set to the maximum value of its acceptable range plus 0.1. If the status is changed to NO, no value for that data type will be inserted into the SHIP database, and a MISSING status value will be inserted. Changing the status back to OK allows the incoming or recorded value entering the DSS to again be inserted into the SHIP database.

It should be noted that the actual incoming or recorded data values entering the DSS may also be out of acceptable data ranges. In these cases, when OK is pushed, the actual out-of-range data value and the DSS-assigned status value will be entered into the SHIP database. The LO, HI, and NO buttons on the DSS data display screens are therefore data override buttons. The status and data values inserted into the SHIP database remain changed for that data type until another button associated with the measurement is clicked.

Inserting Data into the SHIP Database

Once the DSS database structure containing the converted sensor measurement, calculated values, and data status values has been created, the information is inserted into the SHIP database. The DSS actively seeks to insert data into a SHIP database whenever the DSS is operating and the control panel screen is displayed. If a SHIP component is on-line at this time, new data records will be inserted into the SHIP database as quickly as the DSS can receive and process them. Presently this capability is faster than one record per second; however, aboard the *Essayons*, the ship's data logging computer sends the DSS data records every 10 sec; therefore, data are inserted into the SHIP database at the same frequency.

Data insertion into the SHIP database is performed by the DSS using a database interface program named ***Database***. Activating the interface calls a stored procedure, named ***Insert_Dredging_Data***, which resides within the Sybase (SHIP) database. This stored procedure transfers the contents of the DSS database structure to the SHIP database and stores them within the ***Dredging_Data*** table.

For more information concerning this DSS-to-SHIP database interface, see Chapter 6 ("System Interfaces") within this manual. A complete description of the SHIP database tables, their content, and structure is contained within Chapter 4 ("System Database") of this manual.

4 System Database

Database Engine

The Silent Inspector system's central database resides within the SHIP and SHORE components. The database structure in each component is identical.

The system's database is built and accessed with the Sybase relational DBMS, which is commercially available for the Unix operating system. This is the database engine around which the Silent Inspector system has been built. The Sybase DBMS works on a client-server basis with all programs with which it interacts. The database is designed so that multiple functions may (and do) occur simultaneously. Such functions include:

Insertion of data into the database by the DSS
Displaying the newly inserted data for the user
Calculating additional data, such as TDM and dredge state, from inserted data
Copying data into other appropriate database tables
Calculating/summing the time certain dredge activities occur over a load, day, or project
Displaying recorded data as requested by the user
Conducting portions of the data backup and archiving functions

This chapter describes how the Sybase DBMS has been specifically structured to perform the Silent Inspector operations.

Database Structure

The SHIP database consists of a number of data tables and stored procedures. Two types of tables exist within the database: user tables and system tables. User tables are those tables which accept or handle either information input by the user through the system's setup function, or sensor and calculated data and data status values inserted by the DSS or calculated within the SHIP component. System tables are ones that are built, updated, and maintained automatically by Sybase. They contain information on the data contained within the DBMS. For example, there is a system table that contains information about all of the tables, both user and system, known to Sybase. Stored procedures are essentially Sybase subroutines stored within the DBMS which are called by the system's central program code to perform certain operations on the database. The central program code, which controls the automatic processing of the newly inserted data and activates the necessary subroutines and stored procedures to calculate additional parameters (such as the load number, TDM, and the dredge state), store the newly computed values, and trigger automatic reports, is called the real time kernel (***RT kernel***).

There are currently 21 user tables, 13 system tables, and 21 stored procedures within version 1.0 of the SHIP database.

Data Flow Overview

Figure 1 provides an overview of the movement of data through the Silent Inspector system. The system has four primary data input or transfer interfaces, identified in Figure 1 with the letters *a*, *b*, *c*, and *d.* Interfaces a and c are between computers, whereas interfaces b and d are user interactions with the DSS and SHIP components which allow the user to input data on certain parameters into the system via graphical user interface screens within the DSS data entry panel, and the SHIP setup module. The output of data contained within the database user tables to a SHORE component using the system's archive software module is not depicted. The parameters which are transferred over the computer interfaces (a and c) and specifications for those interfaces, are described in detail in Chapter 6, "System Interfaces."

In Figure 1, rectangles with curved ends represent user tables within the database, rectangles with pointed ends represent software modules, true rectangles within the SHIP component box represent computations or actions performed by ***RT Kernel***, and rectangles with wavy bottoms represent reports or graphical displays. Diamonds within the DSS box indicate computations occurring within the DSS. Note that the user must enter much of the project and dredge-related information via the Setup module, whereas the measured sensor data come from the DSS. ***RT Kernel*** is the systems central software program which controls the processing and routing of all incoming data. The flow of data through the user tables can be traced in more detail using Table 1 contained within the next section, "User Tables."

User Tables

Entries into the user tables come from one of three main sources: the user through the setup or downtime modules, insertion by the DSS (***Dredging_Data*** table only), or from computations or data transfers controlled within ***RT Kernel.*** Data values contained within the tables are called data elements herein. Each table can be thought of as a spreadsheet, where each data element is a column within the spreadsheet, and each new entry of values for the data elements fills a row. Each row represents data associated with some time or event (such as a change in the dredging state). The user table names, type of

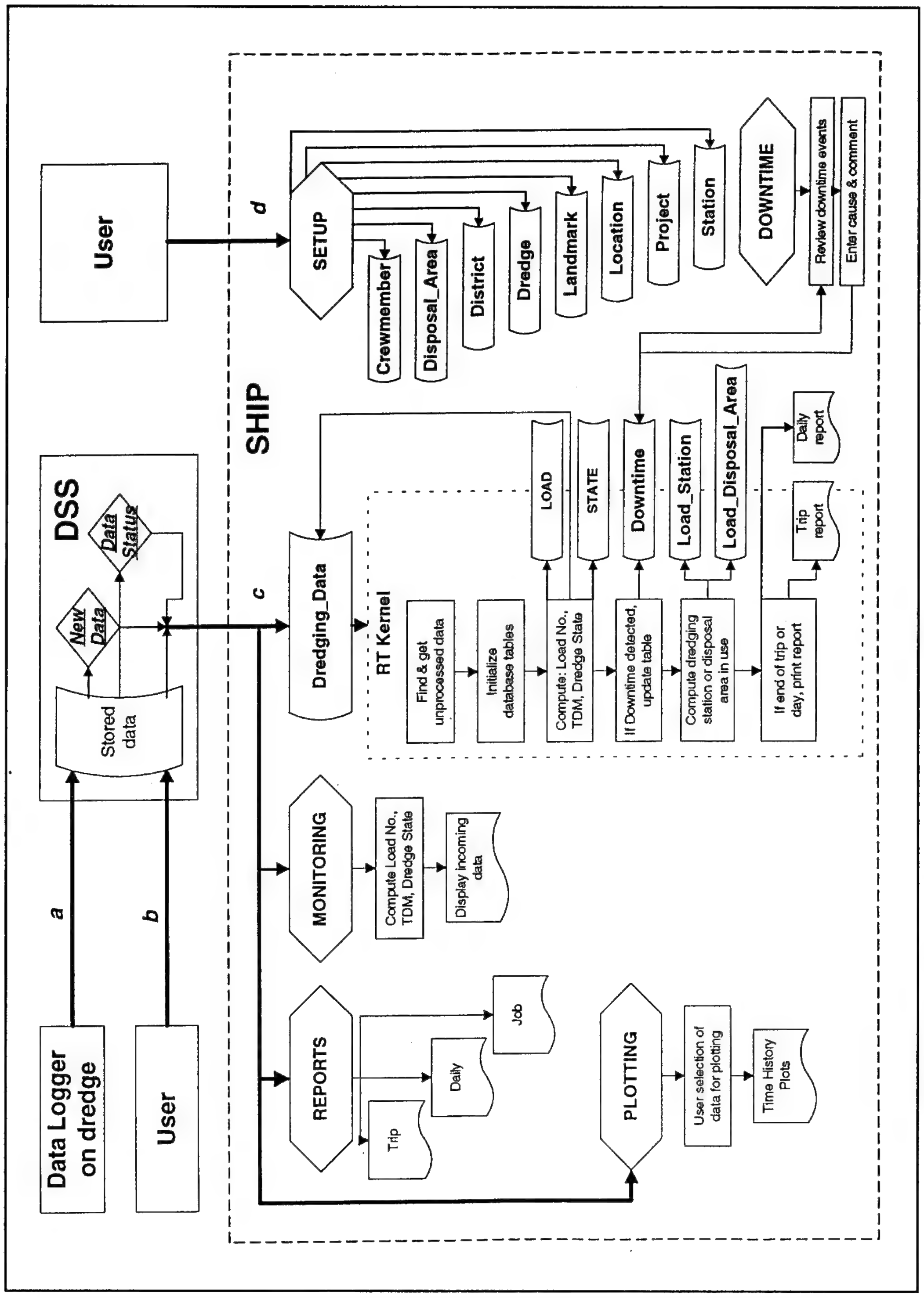


Figure 1. Overview of data flow within DOSIS

data in their contents, and primary data input sources are summarized as follows:

Table Name	Table Contents	Input Source
Crewmember	Names, rank, and ID	Setup module
Disposal_Area	Names, boundary coordinates	Setup module
District	Names, abbreviations	Setup module
Dredge	Vessel information	Setup module
Landmark	Names and coordinates	Setup module
Location	Name	Setup module
Project	Names, project information	Setup module
Station	Dredging area names, boundary coordinates	Setup module
Dredging_Data	Measured data and data status	DSS, RT Kernel
Dredge_Crew	Assigns crewmember to project and dredge	RT Kernel
Downtime	Downtime events and causes/annotation	RT Kernel/Downtime module
Load_Disposal_Area	Assigns a trip (load) to a disposal area	RT Kernel
Load_Station	Assigns a trip (load) to a dredging area	RT Kernel
Load_Table	Trip report summary information	RT Kernel
Location_Station	Assigns a location to a dredging area	RT Kernel
Project_Disposal_Area	Assigns a disposal area to a project	RT Kernel
Project_Dredge	Assigns a dredge to a project, sets limits	RT Kernel
Project_Landmark	Assigns a landmark to a project	RT Kernel
Project_Station	Assigns a dredging area (station) to a project	RT Kernel
Project Summary	Job-to-date report information	Job Report module
State	Trip data for whenever a state change occurs	RT Kernel

The individual data elements contained within each user table are shown in Table 1 on the following pages. Shown in this table are the user table name, purpose, specific data elements contained within each table, units required for the data elements, maximum character length, null condition, input source, and output destinations.

The input source listed in these user tables is the location from which the data value or information for each data element is obtained when it is inserted into that particular table. For instance, the Dredge (name), Project (name), and Contract ID are initially inserted into the DSS component of the system by the user. The DSS inserts this information, along with required sensor data and status values, into the ***Dredging_Data*** table within the SHIP database; therefore, the DSS is the source of these values for that table. These values are then distributed (copied) to 12 other tables when values for other data elements are also inserted into those tables. The ***Dredging_Data*** table, or another table, then becomes the input source for these data elements. When the user is the input source via the setup or downtime modules, those modules are listed under the input source column. RT kernel is the source of values computed within the SHIP component.

While values for each data element can only have one input source from which they are inserted into a user table, the values within a user table may be sent or copied to multiple destinations. The Dredge (name), Project (name), and Contract ID data elements within the ***Dredging_Data*** table are good examples of this. Each is sent from the ***Dredging_Data*** table to several other tables. These three data elements make up the unique identifier for recalling

Table 1
Contents of SHIP Database User Tables

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: CREWMEMBER **Purpose: List crewmember names, rank, and ID**					
FIRST_NAME	text	32	no	Setup	None
LAST_NAME	text	32	no	Setup	None
RANK	text	7	no	Setup	None
ID	text	4	no	Setup	None
Table: DISPOSAL_AREA **Purpose: List disposal areas and their boundaries; used by RT Kernel to determine dredging/disposal in/out of areas**					
NAME	text	32	no	Setup	RT Kernel (Check_Disposal_Areas computation)
BOUNDARY_01_X	feet (Lambert)	8	no	Setup	RT Kernel (Check_Disposal_Areas computation)
BOUNDARY_01_Y	feet (Lambert)	8	no	Setup	RT Kernel (Check_Disposal_Areas computation)
BOUNDARY_02_X	feet (Lambert)	8	no	Setup	RT Kernel (Check_Disposal_Areas computation)
BOUNDARY_02_Y	feet (Lambert)	8	no	Setup	RT Kernel (Check_Disposal_Areas computation)
BOUNDARY_03_X	feet (Lambert)	8	no	Setup	RT Kernel (Check_Disposal_Areas computation)
BOUNDARY_03_Y	feet (Lambert)	8	no	Setup	RT Kernel (Check_Disposal_Areas computation)
BOUNDARY_04_X	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Disposal_Areas computation)
BOUNDARY_04_Y	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Disposal_Areas computation)
Table: DISTRICT **Purpose: Names and abbreviations of Corps of Engineers Districts**					
NAME	text	32	no	Setup	None
ABBREVIATION	text	3	no	Setup	None

(Sheet 1 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: DOWNTIME **Purpose: List and annotate downtime events**					
CONTRACT_ID	text	32	no	Dredging_Data	None
PROJECT (name)	text	32	no	Dredging_Data	None
DREDGE (name)	text	32	no	Dredging_Data	None
START_DATE_TIME	Sybase	8	no	RT Kernel	None
END_DATE_TIME	Sybase	8	no	RT Kernel	None
LOAD_NO	integer	4	no	RT Kernel	None
CAUSE	text	37	no	Downtime	None
COMMENT	text	255	yes	Downtime	None
Table: DREDGE **Purpose: Information on dredges (continued)**					
NAME	text	32	no	Setup	None
DREDGE_TYPE (type of dredge)	text	4	no	Setup	None
OWNERS (names)	text	32	no	Setup	None
YEAR_CONSTRUCTED	integer	4	no	Setup	None
BEAM	feet	8	no	Setup	None
LENGTH	feet	8	no	Setup	None
HORSEPOWER		8	no	Setup	None
DRAFT_FORE_DISTANCE	feet	8	no	Setup	None

(Sheet 2 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: DREDGE **Purpose: Information on dredges (concluded)**					
DRAFT_AFT_DISTANCE	feet	8	no	Setup	None
DRAFT_LIGHT	feet	8	no	Setup	None
DRAFT_LOADED	feet	8	no	Setup	None
HOPPER_FORE_DISTANCE	feet	8	no	Setup	None
HOPPER_AFT_DISTANCE	feet	8	no	Setup	None
HOPPER_LOAD_CAPACITY	long tons	8	no	Setup	None
HOPPER_VOLUME_CAPACITY	cubic yards	8	no	Setup	None
EMPTY_DISPLACEMENT	long tons	8	no	Setup	None
MAX_DISPLACEMENT	long tons	8	no	Setup	None
SPEED_LIGHT	knots	8	no	Setup	None
SPEED_LOADED	knots	8	no	Setup	None
Table: DREDGE_CREW **Purpose: Information on crew of dredge**					
CONTRACT_ID	text	32	no	Setup	None
PROJECT (name)	text	32	no	Setup	None
DREDGE (name)	text	32	no	Setup	None
ID	text	4	no	Setup	None

(Sheet 3 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: DREDGING_DATA **Purpose: Accept data from DSS (continued)**					
CONTRACT_ID	text	32	no	DSS	Multiple tables
PROJECT (name)	text	32	no	DSS	Multiple tables
DREDGE (name)	text	32	no	DSS	Multiple tables
DATE_TIME	Sybase	8	no	DSS	Multiple tables
VESSEL_X	feet (Lambert)	8	yes	DSS	RT Kernel, State table
VESSEL_X_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
VESSEL_Y	feet (Lambert)	8	yes	DSS	RT Kernel, State table
VESSEL_Y_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
VESSEL_DRAFT_FORWARD	feet	8	yes	DSS	RT Kernel, State table
VESSEL_DRAFT_FORWARD_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
VESSEL_DRAFT_AFT	feet	8	yes	DSS	RT Kernel, State table
VESSEL_DRAFT_AFT_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
VESSEL_SPEED	knots	8	yes	DSS	RT Kernel, State table
VESSEL_SPEED_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
VESSEL_HEADING	degrees True	8	yes	DSS	RT Kernel, State table
VESSEL_HEADING_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
VESSEL_COURSE	degrees True	8	yes	DSS	RT Kernel
VESSEL_COURSE_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
DRAGHEAD_DEPTH_PORT	feet	8	yes	DSS	RT Kernel, State table
DRAGHEAD_DEPTH_PORT_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel

(Sheet 4 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: DREDGING_DATA **Purpose: Accept data from DSS (continued)**					
DRAGHEAD_DEPTH_STBD	feet	8	yes	DSS	RT Kernel, State table
DRAGHEAD_DEPTH_STBD_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
HOPPER_LEVEL_FORWARD	feet	8	yes	DSS	RT Kernel
HOPPER_LEVEL_FORWARD_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
HOPPER_LEVEL_AFT	feet	8	yes	DSS	RT Kernel
HOPPER_LEVEL_AFT_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
HOPPER_VOLUME	cubic yards	8	yes	DSS	RT Kernel
HOPPER_VOLUME_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
HOPPER_ULLAGE	feet	8	yes	DSS	RT Kernel
HOPPER_ULLAGE_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
WATER_DEPTH	feet	8	yes	DSS	RT Kernel
WATER_DEPTH_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
TIDE	feet	8	yes	DSS	RT Kernel, State table
TIDE_STATUS	Ok, Lo, Hi, No	2	no	DSS	RT Kernel
HOPPER_DOOR_OPEN	boolean	7	no	DSS	RT Kernel
PUMP_ON_PORT	boolean	7	no	DSS	RT Kernel, State table
PUMP_ON_STBD	boolean	7	no	DSS	RT Kernel, State table
PUMP_MATERIAL_PORT	boolean	7	no	DSS	RT Kernel (dredge state calculation)
PUMP_MATERIAL_STBD	boolean	7	no	DSS	RT Kernel (dredge state calculation)
PUMP_OUT_ON	boolean	7	no	DSS	RT Kernel (dredge state calculation)

(Sheet 5 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: DREDGING_DATA **Purpose: Accept data from DSS (concluded)**					
LOAD_NO	integer	4	yes	RT Kernel	Load table
STATE	text	7	yes	RT Kernel	RT Kernel, State table
TDM	long tons	8	yes	RT Kernel	Daily, Job reports
Table: LANDMARK **Purpose: Information on marine landmarks**					
NAME	text	32	no	Setup	None
POSITION_X	feet (Lambert)	8	no	Setup	None
POSITION_Y	feet (Lambert)	8	no	Setup	None
Table: LOAD_DISPOSAL_AREA **Purpose: Assign a disposal area to each load number on a project**					
CONTRACT_ID	text	32	no	Dredging_Data	None
PROJECT (name)	text	32	no	Dredging_Data	None
DREDGE (name)	text	32	no	Dredging_Data	None
LOAD_NO	integer	4	no	RT Kernel	None
DISPOSAL_AREA (name)	text	32	no	Disposal_Area	None
DISPOSAL_START_DATE_TIME	Sybase	8	no	RT Kernel	None
Table: LOAD_STATION **Purpose: Assign a station (dredging area) to each load number on a project (continued)**					
CONTRACT_ID	text	32	no	Dredging_Data	None
PROJECT (name)	text	32	no	Dredging_Data	None
DREDGE (name)	text	32	no	Dredging_Data	None

(Sheet 6 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: LOAD_STATION **Purpose: Assign a station (dredging area) to each load number on a project (concluded)**					
LOAD_NO	integer	4	no	RT Kernel	None
STATION_AREA (name)	text	32	no	Station	None
STATION_START_DATE_TIME	Sybase	8	no	RT Kernel	None
Table: LOAD_TABLE **Purpose: Tabulation of trip report information**					
CONTRACT_ID	text	32	no	Dredging_Data	Trip Report
PROJECT (name)	text	32	no	Dredging_Data	Trip Report
DREDGE (name)	text	32	no	Dredging_Data	Trip Report
LOAD_NO	integer	4	no	RT Kernel	Trip Report
DATE	Sybase	8	no	Dredging_Data	Trip Report
PUMPING_TIME	minutes	8	no	RT Kernel	Trip Report
TURNING_TIME	minutes	8	no	RT Kernel	Trip Report
SAILING_FULL_TIME	minutes	8	no	RT Kernel	Trip Report
DUMPING_TIME	minutes	8	no	RT Kernel	Trip Report
SAILING_EMPTY_TIME	minutes	8	no	RT Kernel	Trip Report
DOWN_TIME	minutes	8	no	RT Kernel	Trip Report
TOTAL_TIME	minutes	8	no	RT Kernel	Trip Report
TDM	long tons	8	no	RT Kernel	Trip Report
COMMENT	text	255	yes		Trip Report

(Sheet 7 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: LOCATION **Purpose: Assign a name to each station (dredging area)**					
NAME	text	32	no	Setup	None
Table: LOCATION_STATION **Purpose: Assign a location name to each station listed in the Station table**					
LOCATION (name)	text	32	no	Location	None
STATION (name)	text	32	no	Station	None
Table: PROJECT **Purpose: Basic Project Information**					
CONTRACT_ID	text	32	no	Setup	None
NAME	text	32	no	Setup	None
DISTRICT	text	3	no	Setup	None
THE_TYPE	text	13	no	Setup	None
CONTRACTOR	text	32	yes	Setup	None
START_DATE	Sybase	8	yes	Setup	None
FINISH_DATE	Sybase	8	yes	Setup	None
Table: PROJECT_DISPOSAL_AREA **Purpose: Assign a disposal area to a project name/contract ID pairing**					
CONTRACT_ID	text	32	no	Dredging_Data	RT Kernel (Check_Disposal_Area computation)
PROJECT (name)	text	32	no	Dredging_Data	RT Kernel (Check_Disposal_Area computation)
DISPOSAL_AREA	text	32	no	Disposal_Area	RT Kernel (Check_Disposal_Area computation)

(Sheet 8 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: PROJECT_DREDGE **Purpose: Assign a dredge and dredge properties to a project name/contract ID pair**					
CONTRACT_ID	text	32	no	Dredging_Data	None
PROJECT (name)	text	32	no	Dredging_Data	None
DREDGE (name)	text	32	no	Dredging_Data	None
CAPTAIN	text	4	yes	Setup	None
SPEED_TOLERANCE	knots	8	yes	Setup	None
DEPTH_TOLERANCE	feet	8	yes	Setup	None
DRAFT_LIMIT	feet	8	yes	Setup	None
DRAG_LIMIT	feet	8	yes	Setup	None
TRIM_LIMIT	feet	8	yes	Setup	None
TURN_LIMIT	degrees/min	8	yes	Setup	None
DRY_MATERIAL_MASS_DENSITY	grams/liter	8	yes	Setup	RT Kernel (TDM calculation)
WATER_MASS_DENSITY	grams/liter	8	yes	Setup	RT Kernel (TDM calculation)
DREDGE_AIDS	text	255	yes	Setup	None
COMMENT	text	255	yes	Setup	None
Table: PROJECT_LANDMARK **Purpose: Assign a marine landmark to a project name/contract ID pair**					
CONTRACT_ID	text	32	no	Dredging_Data	None
PROJECT (name)	text	32	no	Dredging_Data	None
LANDMARK (name)	text	32	no	Landmark	None

(Sheet 9 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: PROJECT_STATION **Purpose: Assign a station (dredging area) to a project name/contract ID pair**					
CONTRACT_ID	text	32	no	Dredging_Data	RT Kernel (Check_Station computation)
PROJECT (name)	text	32	no	Dredging_Data	RT Kernel (Check_Station computation)
STATION (name)	text	32	no	Station	RT Kernel (Check_Station computation)
Table: PROJECT_SUMMARY **Purpose: Tabulation of job report information (continued)**					
CONTRACT_ID	text	32	no	Dredging_Data	Job Report
PROJECT (name)	text	32	no	Dredging_Data	Job Report
DREDGE (name)	text	32	no	Dredging_Data	Job Report
OPENING_DATE	Sybase	8	no	RT Kernel	Job Report
CLOSING_DATE	Sybase	8	yes	RT Kernel	Job Report
PUMPING_TIME	minutes	8	no	RT Kernel	Job Report
TURNING_TIME	minutes	8	no	RT Kernel	Job Report
SAILING_FULL_TIME	minutes	8	no	RT Kernel	Job Report
DUMPING_TIME	minutes	8	no	RT Kernel	Job Report
SAILING_EMPTY_TIME	minutes	8	no	RT Kernel	Job Report
NON_EFFECTIVE_TIME	minutes	8	no	RT Kernel	Job Report
LOST_TIME	minutes	8	no	RT Kernel	Job Report
TO_BE_DEFINED_TIME	minutes	8	no	RT Kernel	Job Report
FUEL_AND_SUPPLIES_TIME	minutes	8	no	RT Kernel	Job Report
WHARF_OR_ANCHORAGE_TIME	minutes	8	no	RT Kernel	Job Report
OPPOSING_NATURAL_ELEMENTS_TIME	minutes	8	no	RT Kernel	Job Report

(Sheet 10 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: PROJECT_SUMMARY **Purpose: Tabulation of job report information (concluded)**					
TRAFFIC_AND_BRIDGES_TIME	minutes	8	no	RT Kernel	Job Report
MINOR_OPERATING_REPAIRS_TIME	minutes	8	no	RT Kernel	Job Report
TRANSFER_BETWEEN_WORKS_TIME	minutes	8	no	RT Kernel	Job Report
LAY_TIME	minutes	8	no	RT Kernel	Job Report
FIRE_AND_BOAT_DRILLS_TIME	minutes	8	no	RT Kernel	Job Report
MISCELLANEOUS_TIME	minutes	8	no	RT Kernel	Job Report
MAJOR_REPAIRS_TIME	minutes	8	no	RT Kernel	Job Report
CESSATION_TIME	minutes	8	no	RT Kernel	Job Report
COLLISIONS_TIME	minutes	8	no	RT Kernel	Job Report
ESTIMATED_COST	dollars	8	no	Setup	Job Report
JOB_TO_DATE_COST	dollars	8	no	Setup	Job Report
COST_PER_MINUTE	money	8	no	RT Kernel	Job Report
TONS_RETAINED	long tons	8	no	RT Kernel	Job Report
Table: STATE **Purpose: Tabulation of values at the start of each new dredging state, for use in the trip report (continued)**					
CONTRACT_ID	text	32	no	Dredging_Data	Trip Report
PROJECT (name)	text	32	no	Dredging_Data	Trip Report
DREDGE (name)	text	32	no	Dredging_Data	Trip Report
LOAD_NO	integer	4	no	RT Kernel	Trip Report
START_DATE_TIME	Sybase	8	no	Dredging_Data	Trip Report
VESSEL_X	feet (Lambert)	8	yes	Dredging_Data	Trip Report

(Sheet 11 of 13)

Table 1 (Continued)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: STATE **Purpose: Tabulation of values at the start of each new dredging state, for use in the trip report (concluded)**					
VESSEL_Y	feet (Lambert)	8	yes	Dredging_Data	Trip Report
VESSEL_HEADING	degrees True	8	yes	Dredging_Data	Trip Report
VESSEL_SPEED	knots	8	yes	Dredging_Data	Trip Report
VESSEL_DRAFT_FORWARD	feet	8	yes	Dredging_Data	Trip Report
VESSEL_DRAFT_AFT	feet	8	yes	Dredging_Data	Trip Report
PORT_PUMP_STATUS	boolean	7	no	Dredging_Data	Trip Report
DRAGHEAD_DEPTH_PORT	feet	8	yes	Dredging_Data	Trip Report
DRAGHEAD_ELEVATION_PORT	feet	8	yes	Dredging_Data	Trip Report
STBD_PUMP_STATUS	boolean	7	no	Dredging_Data	Trip Report
DRAGHEAD_DEPTH_STBD	feet	8	yes	Dredging_Data	Trip Report
DRAGHEAD_ELEVATION_STBD	feet	8	yes	Dredging_Data	Trip Report
TIDE	feet	8	yes	Dredging_Data	Trip Report
STATE	text	7	no	RT Kernel	Trip Report
Table: STATION **Purpose: Assigns boundary coordinates to a station (dredging area) (continued)**					
NAME	text	32	no	Setup	RT Kernel (Check_Station computation)
BOUNDARY_01_X	feet (Lambert)	8	no	Setup	RT Kernel (Check_Station computation)
BOUNDARY_01_Y	feet (Lambert)	8	no	Setup	RT Kernel (Check_Station computation)
BOUNDARY_02_X	feet (Lambert)	8	no	Setup	RT Kernel (Check_Station computation)
BOUNDARY_02_Y	feet (Lambert)	8	no	Setup	RT Kernel (Check_Station computation)
BOUNDARY_03_X	feet (Lambert)	8	no	Setup	RT Kernel (Check_Station computation)

(Sheet 12 of 13)

Table 1 (Concluded)

Data Elements	Units	Length	Nulls	Input Source	Output Destinations
Table: STATION Purpose: Assigns boundary coordinates to a station (dredging area) (concluded)					
BOUNDARY_03_Y	feet (Lambert)	8	no	Setup	RT Kernel (Check_Station computation)
BOUNDARY_04_X	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_04_Y	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_05_X	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_05_Y	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_06_X	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_06_Y	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_07_X	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_07_Y	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_08_X	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_08_Y	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_09_X	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_09_Y	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_10_X	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)
BOUNDARY_10_Y	feet (Lambert)	8	yes	Setup	RT Kernel (Check_Station computation)

(Sheet 13 of 13)

the data related to any specific project. ***RT Kernel*** controls this movement or flow of data between the user tables.

System Tables

The second type of tables within the SHIP database are called system tables. As mentioned before, system tables are ones that are built, updated, and maintained automatically by Sybase. Neither the user nor the system's Ada code directly accesses or modifies the contents of the System tables. These tables contain "meta-data," which are data about the data or tables within the DBMS. The system tables presently contained within the database are:

Table Name
sysalternates
syscolumns
syscomments
sysdepends
sysindexes
syskeys
syslogs
sysobjects
sysprocedures
sysprotects
syssegments
systypes
sysusers

A complete description of all the system tables and their purpose is contained within the Sybase DBMS manuals.

Stored Procedures

While ***RT Kernel*** controls the overall flow of data between the user tables, the programs which perform the actual input, retrieval, or updating of data within the user tables are called stored procedures. These procedures are essentially Sybase subroutines called by ***RT Kernel*** or other software modules, such as the DSS, when it inputs data into the ***Dredging_Data*** table in the SHIP database. When invoked by name, Sybase fetches the appropriate stored procedure and executes the operations therein independent from and without intervention by the calling program. Input parameters are passed to the stored procedure when it is invoked, and output parameters are returned by the stored procedure when it completes its processing. The procedures are stored within the Sybase DBMS. These procedures, while written specific to the Silent Inspector system, were developed under the same protocols as standard Sybase stored procedures.

The stored procedures perform one of the following three operations:

INSERT:	Input data into a user table in the database.
SELECT:	Retrieve data from a user table in the database.
UPDATE:	Update data within a user table in the database.

There are 21 stored procedures within the SHIP database. The name and purpose of each procedure are listed below:

Procedure Name	**Procedure Purpose**
INSERT_DOWNTIME_DATA	Insert a downtime event into the Downtime table
INSERT_DREDGING_DATA	Insert data from the DSS into the Dredging Data table
INSERT_LOAD_DATA	Insert summary data for a load into the Load table
INSERT_LOAD_DISPOSAL_AREA	Insert a dump event for a disposal area into the Load Disposal Area table
INSERT_LOAD_STATION	Insert a dredging event for a station (site) into the Load Station table
INSERT_PROJECT_DREDGE	Insert information on a dredge assigned to a project into the Project Dredge table
INSERT_PROJECT_SUMMARY_DATA	Insert project summary data into the Project Summary Data table
INSERT_STATE	Insert information related to a change in the dredge state (activity) into the State table
SELECT_DISPOSAL_AREA_DISTINCT	Retrieve a unique disposal area name from the Disposal Area Table based on its boundary data
SELECT_DOWNTIME_DISTINCT	Retrieve a unique downtime event having a user specified Contract ID, Project name, Dredge name, and Start_Date_Time from the Downtime table
SELECT_LATEST_DREDGING_DATA	Retrieve the data record from the Dredging Data table having the latest date/time value
SELECT_LOAD_DISTINCT	Retrieve the data from the Load table for a user-specified Contract ID, Project name, Dredge name, and Load Number
SELECT_MIN_MAX_	Retrieve the minimum and maximum date/time values that exist in data records in the Dredging Data table
SELECT PROCESSED_DREDGING_DATA	Retrieve the latest processed (been assigned a dredge state by the RT Kernel program) data record in the Dredging Data table
SELECT_PROJECT_DISTINCT	Retrieve a unique project data record having a user-specified Contract ID, and Project name from the Project table, and retrieve the District name from the District table
SELECT_PROJ_SUMMARY_DISTINCT	Retrieve a unique project summary data record from the Project Summary table having a user-specified Contract ID, Project name, and Dredge name
SELECT_STATION_DISTINCT	Retrieve a unique station (dredging area) name from the Station table based on its boundary data
SELECT_UMPROC_DREDGING_DATA	Retrieve the earliest unprocessed (not been assigned a dredge state by the RT Kernel program) data record in the Dredging Data table
UPDATE_DOWNTIME	Update a downtime event in the Downtime table with a user input cause and comment
UPDATE_DREDGING_DATA	Update a dredging data record in the Dredging Data table with a load number, dredge state, and TDM value
UPDATE_PROJECT_SUMMARY	Update a project summary data record in the Project Summary table

Stored procedures are procedures created specifically for use by Sybase. The stored procedures are created and changed using a text editor, and entered into the Sybase DBMS using isql. They are defined to Sybase by name and may have a parameter list, just like other procedures in other languages. They are invoked (called) by name from within ***RT Kernel*** or the other Ada code modules within the system, and parameters are then supplied to match the parameter list of the stored procedure.

The specific code associated with each of these stored procedures is provided in Appendix A. Each data element within the data tables that is input, retrieved, or updated by these stored procedures is listed within this code.

RT Kernel

The system's central program within the SHIP component, which processes newly arriving data and controls the overall flow of data, is called ***RT Kernel.*** ***RT Kernel*** starts running when the user starts the SHIP component of the Silent Inspector system by clicking on the DOSIS icon. ***RT Kernel*** then runs continuously whether or not the DSS is operating and inserting data into the SHIP database. ***RT Kernel*** has several functions:

a. Looks at each data record inserted by the DSS into the ***Dredging_Data*** table (called unprocessed data) and computes the dredge state (activity) occurring at that time, the tons dry weight of material collected in the hopper at that time (TDM), and assigns a load number to the data record. Once these three values have been attached to the data within the ***Dredging_Data*** table, the data are considered processed data.

b. If the dredge state is determined to be "Down," ***RT Kernel*** makes an entry in the ***DOWNTIME*** table.

c. Distributes the processed data to other database tables.

d. Computes and stores data for load and daily reports.

e. Determines when the end of a trip or day has occurred, and initiates Unix processes (subroutines), which create a trip or daily report.

f. Determines whether or not dredging areas (stations) and disposal areas related to the current project have been entered into the ***PROJECT_STATION*** and ***PROJECT_DISPOSAL_AREA*** tables; and if so, determines whether or not the vessel is dredging or disposing in those defined areas; and if so, enters that fact into the ***LOAD_STATION*** and ***LOAD_DISPOSAL_AREA*** tables.

To perform these functions, ***RT Kernel*** calls upon several subroutines. Those subroutines in turn utilize the Sybase stored procedures to access or insert data within the database tables. The steps ***RT Kernel*** performs and the names of the subroutines utilized are summarized below in order of their performance within ***RT Kernel***. Steps that are indented represent steps which occur within the called subroutines.

Steps Performed	**Subroutine Names**
Log on to DOSIS database upon SHIP module startup	Database.Signon
Initialize data monitoring module from last processed data	Initialize_Data_Monitoring
Retrieve latest processed data record	Database.Select_Processed _Dredging_Data
Retrieve earliest unprocessed data record from database	Get_Unprocessed_Dredging_Data; & Database. Select_Unprocessed Dredging_Data
If data are for a different project, initialize project data	Initialize_Project_Data
Identify disposal areas assigned to the project	Initialize_Disposal_Areas
Identify dredging areas assigned to the project	Initialize_Stations
Compute time between new and previous data records	Elapsed_Time
Compute the dredge state for the new data record	SI_Computations.Compute_Dredge_State
If data are for a new load (trip), initialize a new load record	Initialize_Load
Update Load table for elapsed time in new dredge state	Load_State_Time
Update several database tables with new record data	Kernel_Database_Transaction
Check for the end of a load	New_State
Add load data to existing load no. in Load table	Add_Load
Create new load if needed	New_Load
Check for a change in the dredge state	New_State
Add dredge state change data to State, Load, and Dredging_Data tables	Add_State_Change
Check for start of downtime state and initialize downtime record	Initialize_Downtime
Check for end of downtime state and enter data	Add_Downtime
Check for and mark start of dumping state	Dumping_Started
Check for dumping in a project disposal area	Check_Disposal_Areas
Add data to Load_Disposal_Area table	Add_Dumping_Entry
Check for dredging in project dredging area (station)	Check_Stations
Add data to Load_Station table	Add_Dredging_Entry
If dumping state is just completed, and to cut started, compute sailing empty displacement	SI_Computations. Dredge_Empty_Displacement
If not sailing empty, compute TDM	SI_Computations.Tonnage_Dry_Measure
Update Dredging_Data table with state & load no.	Database.Update_Dredging_Data
Complete the database transaction	Database.Commit_Transaction
Print reports as necessary	Print_Manager
Print a load (trip) report	Print_Trip_Report
Print a daily report	Print_Daily_Report
Save current processed data record values for use in processing next unprocessed record	Save_Values

5 System Computations

The DOSIS system performs calculations within both the DSS and SHIP components. No computations are performed within the SHORE component at the present time, as all necessary values have been computed and stored in the DOSIS database prior to its archiving and transfer to a SHORE unit.

DSS Component Calculations

Calculations within the DSS component are not necessarily identical between DSS units. This is because each DSS is configured to receive and process a specific suite of sensor data being provided by a particular dredge. In all cases, the output of the DSS to the SHIP database is identical in its content, format, and method of interaction with the SHIP component. Therefore, while the DSS calculations may vary, the calculated values and their units will remain the same between DSS units. The training and reference DSS performs the following calculations:

Calculation Name	**Units**	**Purpose**
Average hopper ullage	feet	Calculate the average of the four ullage sensor values
Average hopper level forward	feet	Determine the level of the water in the hopper at the forward end
Average hopper level aft	feet	Determine the level of the water in the hopper at the aft end
Hopper volume	yds^3	Determine the volume of water and material within the hopper based upon the fore and aft water levels
Port/Stbd pump on/off	on/off	Determine if the port and starboard pumps are on or off
Port/Stbd material recovery	yes/no	Determine if material is being recovered through the port and starboard drag arms
Status of parameter values	Ok, Lo, Hi, No	Determine if the received or computed data values are within or beyond acceptable limits, as defined in a data acceptable range table, or if the data are missing

Each of the DSS calculations are provided below. The data parameters used and computed in each calculation, their abbreviation (if any) used in the equations, their units, and the source of the values used for the parameters are presented first. The source of a parameter can either be a SHIP database table, another computation, or a fixed value based on a particular dredge. The exact names of the parameters as they appear within the database tables or within the equations are shown. The detailed equations are then listed.

Average hopper ullage

This computation is a straight average of the four hopper ullage readings.

Parameters Used and Sources:

Parameter	Units	Data Source
Ullage meter No. 1	feet	DSS
Ullage meter No. 2	feet	DSS
Ullage meter No. 3	feet	DSS
Ullage meter No. 4	feet	DSS
Average_Ullage	feet	Computed

Equations:

Average_Ullage = (Ullage 1 + Ullage 2 + Ullage 3 + Ullage 4) / 4

Average hopper level forward

This computation converts the forward ullage readings into an average water level in the hopper at its forward end.

Parameters Used and Sources:

Parameter	Units	Data Source
Ullage meter No. 1	feet	DSS
Ullage meter No. 2	feet	DSS
Average_Ullage_Forward	feet	Computed
Hopper_Bottom_To_Sensors	feet	Fixed; specific to Essayons

Equations:

Average_Ullage_Forward = (Ullage 1 + Ullage 2) / 2
Average_Level_Forward = Hopper_Bottom_To_Sensors - Average_Ullage_Forward

Note: This calculation is currently not performed in the training and reference DSS. Instead, this value is set to a default value with a status value of "missing."

Average hopper level aft

This computation converts the aft ullage readings into an average water level in the hopper at its aft end.

Parameters Used and Sources:

Parameter	Units	Data Source
Ullage meter No. 3	feet	DSS
Ullage meter No. 4	feet	DSS
Average_Ullage_Aft	feet	Computed
Hopper_Bottom_To_Sensors	feet	Fixed; specific to Essayons

Equations:

Average_Ullage_Aft = (Ullage 3 + Ullage 4) / 2
Average_Level_Aft = Hopper_Bottom_To_Sensors - Average_Ullage_Aft

Note: This calculation is currently not performed in the training and reference DSS. Instead, it is set to a default value and a status of "missing."

Hopper volume

The *Essayons* hydrostatic curves were used to derive the equations used to compute the hopper volume. The average ullage reading was used to compute the hopper volume as shown below.

Parameters Used and Sources:

Parameter	Units	Data Source
Average_Ullage	feet	Computed
Hopper_Volume	yds^3	Computed
Constants	none	Fixed; specific to Essayons

Equations:

If Average_Ullage <= 11, then
Hopper_Volume = 6852.57 - (150.74067 * Average_Ullage)

otherwise,

If Average_Ullage > 11 <= 31, then
Hopper_Volume = 7534.1 - (209.45 * Average_Ullage)

otherwise,

If Average_Ullage > 31, then
Hopper_Volume = -75179.0 + (9346.8 * Average_Ullage) - (411.27 * Average_Ullage^2) + (7.7651 * Average_Ullage^3) - (0.053733 * Average_Ullage^4)

The constants listed in these equations are derived from the volume curves/equations for the *Essayons* hopper.

Port/starboard pumps on/off

Port and starboard pump on/off determinations are computed based upon the draghead velocity measurements.

Parameters Used and Sources:

Parameter	Units	Data Source
Port_Draghead_Velocity	feet/sec	DSS
Stbd_Draghead_Velocity	feet/sec	DSS
Pump_On_Port	true/false	Computed
Pump_On_Stbd	true/false	Computed

Equations:

If Port_Draghead_Velocity > 10.0, then Pump_On_Port = true ,then,
If Port_Draghead_Velocity ≤ 10.0, then Pump_On_Port = false

otherwise

If Stbd_Draghead_Velocity > 10.0, then Pump_On_Stbd = true, then,
If Stbd_Draghead_Velocity ≤ 10.0, then Pump_On_Stbd = false

Port/starboard material recovery true/false

Port and starboard material recovery determinations are computed based upon draghead density measurements.

Parameters Used and Sources:

Parameter	Units	Data Source
Port_Draghead_Density	g/l	DSS
Stbd_Draghead_Density	g/l	DSS
Pump_Material_Port	true/false	Computed
Pump_Material_Stbd	true/false	Computed

Equations:

If Port_Draghead_Density ≥ 1.05, then Pump_Material_Port = true, then,
If Port_Draghead_Density < 1.05, then Pump_Material_Port = false

otherwise

If Stbd_Draghead_Density ≥ 1.05, then Pump_Material_Stbd = true, then,
If Stbd_Draghead_Density < 1.05, then Pump_Material_Stbd = false

Pumpout pump on/off

Hopper pumpout pump on/off determinations are computed based upon velocity measurements in the discharge pipe. This parameter is not currently included in the parameters supplied *Essayons* data logging computer to the DSS, but is intended as a required parameter.

Parameters Used and Sources:

Parameter	Units	Data Source
Port_Discharge_Pipe_Velocity	feet/sec	DSS
Stbd_Discharge_Pipe_Velocity	feet/sec	DSS
Pumpout_Pump_On_Port	true/false	Computed
Pumpout_Pump_On_Stbd	true/false	Computed

Equations:

If Port_Discharge_Pipe_Velocity > 10.0, then Pumpout_Pump_On_Port = true ,then,
If Port_Discharge_Pipe_Velocity $\leq$ 10.0, then Pumpout_Pump_On_Port = false

otherwise

If Stbd_Discharge_Pipe_Velocity > 10.0, then Pumpout_Pump_On_Stbd = true, then,
If Stbd_Discharge_Pipe_Velocity $\leq$ 10.0, then Pumpout_Pump_On_Stbd = false

Data status

The status values that can be assigned to each sensor or computed data values are:

Status	Abbr.	Meaning
ACCEPTABLE	OK	sensor measurement is within acceptable limits
OUT_OF_RANGE_LOW	LO	sensor measurement is out-of-range low
OUT_OF_RANGE_HIGH	HI	sensor measurement is out-of-range high
MISSING	NO	sensor measurement is missing

Minimum and maximum acceptable values for each type of data are contained within a file, named ***RANGE.DAT***, that is loaded into the DSS. The values presently set for use with the *Essayons* are shown in Chapter 3. To determine the status of each data value, the DSS compares the values in each data record it receives to the minimum and maximum values in the ***RANGE.DAT*** file. An ADA code subroutine performs this comparison.

SHIP Component Calculations

Calculations within all SHIP components are identical, as the contents of all SHIP databases are identical. Only when data values required for the calculations are missing or out of the acceptable range will the calculations not be performed identically. In most cases the necessary calculations can always be made; however, in cases where a calculation is not possible, the value being calculated is set to a default value, which is normally 0.

The SHIP component performs the following calculations:

Calculation Name	Units	Purpose
Load number	integer	Calculates and assigns a consecutive load number to each load performed on a dredging project, as defined by the project name and contract ID.
Tons dry measure	long tons	Calculates the weight of sediment in the hopper on a dry weight basis assuming an individual sediment particle density of 2,650 g/cm^3.
Dredge state	none	Calculates the activity the dredge is performing at the time of each measurement record (10-sec intervals).
Dredging area	none	Calculates and assigns the name of a dredging location to each load by comparing the dredge's position during dredging of the load with pre-entered dredging area coordinates.
Disposal area	none	Calculates and assigns the name of a disposal area to each load by comparing the dredge's position during dumping of the load with pre-entered disposal area coordinates.
Time summations	hh:mm:ss	Calculates the amount of time the dredge performs various activities during a load, a day, and a job, or other time period defined by the user.

Details of each of the SHIP calculations are provided below. Again, where practical, a graphical overview of the data parameters used and computed from each calculation is presented first. The exact names of the parameters as they appear within the database tables or within the equations are shown. The detailed equations are then listed, followed by a table which lists each parameter, its abbreviation (if any) used in the equations, its units, and the source of the values used for the parameters.

Load number

A new load number is computed after each dumping phase has been completed. Load numbers are computed by incrementing the previous load number by 1. The load number is the highest load number currently in the ***DREDGING_DATA*** table for the current contract ID and dredge name. If no data exist in the ***DREDGING_DATA*** table for the contract ID and dredge name, then the initial load number is set to 1.

Tons dry measure

The TDM value is computed in two steps, as follows:

a. Determine the displacement of the dredge when the hopper is empty.

b. Determine the tonnage dry measure of the material in the hopper.

The detailed computations within each of these steps proceed as follows:

Step 1: Determine the displacement of the dredge when empty.

Parameters Used and Source:

Abbr.	Parameter	Units	Data Source
HV	Hopper_Volume	yd^3	Dredging_Data table
W_e	Empty_Displacement	long tons	Computed
W_t	Total_Displacement	long tons	Computed
D_{cb}	Center_Buoyancy_Draft	feet	Computed
D_f	Vessel_Draft_Forward	feet	Dredging_Data table
D_a	Vessel_Draft_Aft	feet	Dredging_Data table
B_f	Buoyancy_Forward (60.417)	feet	Specific to Essayons
B_a	Buoyancy_Aft (121.917)	feet	Specific to Essayons
SC	Slope_Closed (591.667)	none	Specific to Essayons
CC	constant closed (-2383.333)	none	Specific to Essayons
md_w	Water_Density (1025.0)	g/l	Fixed
C_f	Conversion_Factor (1328.8)	none	Fixed

Values for constants are shown in parentheses after the parameter name. Constants specific to the *Essayons* were obtained from the vessel's specifications. The factor for converting long tons per cubic yard into grams per liter is 1328.8.

Equations:

if HV <= 0.0, then

$$W_e = 8300.0$$

else (otherwise):

$$D_{cb} = D_f + (D_a - D_f) * (B_f / B_a)$$
$$W_t = SC * D_{cb} + CC$$
$$W_e = W_t - (HV * md_w / C_f)$$

Step 2: Determine the tonnage dry measurement of material within the hopper.

Parameters Used and Source:

Abbr.	Parameter	Units	Data Source
HV	Hopper_Volume	yd^3	Dredging_Data table
W_e	Empty_Displacement	long tons	Computed
W_t	Total_Displacement	long tons	Computed
D_{cb}	Center_Buoyancy_Draft	feet	Computed
D_f	Vessel_Draft_Forward	feet	Dredging_Data table
D_a	Vessel_Draft_Aft	feet	Dredging_Data table
B_f	Buoyancy_Forward (60.417)	feet	Specific to Essayons
B_a	Buoyancy_Aft (121.917)	feet	Specific to Essayons
SC	Slope_Closed (591.667)	none	Specific to Essayons
CC	Constant closed (-2383.333)	none	Specific to Essayons
md_w	Water_Density (1025.0)	g/l	Fixed
md_h	Load_Density	g/l	Computed
md_m	Dry_Material_Density (2650.0)	g/l	Fixed
C_f	Conversion_Factor (1328.8)	none	Fixed
TDM	Tonnage_Dry_Measure	long tons	Computed

Equations:

```
if $HV <= 0.0$, then
  $TDM = 0.0$

otherwise:

  $D_{cb} = D_f + (D_a - D_f) * (B_f / B_a)$
  $W_t = SC * D_{cb} + CC$
  $md_h = ((W_t - W_e) / HV) * 1328.8$
  $TDM = ((md_h - md_w) / (md_m - md_w)) * md_m * HV / C_f$
```

The constant values listed above are for the *Essayons*, and are directly coded into the TDM equations utilized within ***RT Kernel***. The **Setup** module will eventually permit the user to enter these values into the **Dredge** table. ***RT Kernel*** will then look these constant values up within the dredge table according to the dredge in use.

Dredge state

For each data record arriving from the DSS, the system calculates the ongoing activity at the time of the measurements. These activities are called dredge states. The system currently assigns one of seven dredging states to each data record. The dredge states are:

a. Sailing empty.

b. Dredging material.

c. Pumping (with no material recovery).

d. Turning.

e. Sailing full.

f. Dumping.

g. Downtime.

Calculating the dredge state is performed in eight steps. These steps are as follows:

a. Compute the rate of change of the vessel heading.

b. Determine the maximum draft for detecting turning.

c. Compute the depth above which the dragheads will be considered up (minimum dredging depth), and no dredging is occurring.

d. Check if either pump is pumping material.

e. Check if both dragheads are raised above the minimum dredging depth.

f. Determine whether or not the vessel is turning.

g. Determine which dredge states are possible.

h. Determine which dredge state is occurring based on the computed possible dredge states (results of item g).

Steps a-f compute information needed to determine which dredge states may exist (Step g). The final selection of a dredge state from the possible dredge states is performed in Step h. The detailed computations within each of the eight steps are shown below.

In these computations, certain values in the equations, such as the Minimum_Sailing_Speed, are fixed within the system. These values were entered during system development and cannot be changed except by a systems engineer. The forward/aft empty and loaded drafts are for the *Essayons*; the remaining values are appropriate for use on any dredge. These parameters, and their current values, are listed below:

Parameter	**Value**	**Units**
Forward_Draft_Loaded	24.0	feet
Forward_Draft_Empty	17.0	feet
Aft_Draft_Loaded	26.0	feet
Aft_Draft_Empty	24.5	feet
Minimum_Turning_Rate	.05	degrees
Minimum_Sailing_Speed	4.0	knots
Minimum_Dredging_Speed	0.1	knots
Minimum_Dumping_Speed	0.1	knots

Step a: Determine a rate of change of the gyro heading.

Parameters Used and Sources:

Parameter	**Units**	**Data Source**
Vessel_Heading	degrees	Dredging_Data table
Previous_Course	degrees	Dredging_Data table
Gyro_Rate	degrees/sec	Computed

Equations:

If Delta_Time > 0.0, then
Gyro_Rate = [(Vessel_Heading - Previous Course) / Delta_Time)]

Step b: Determine the maximum vessel draft of fore and aft readings.

Parameters Used and Sources:

Parameter	**Units**	**Data Source**
Vessel_Draft_Forward	feet	Dredging_Data table
Vessel_Draft_Aft	feet	Dredging_Data table

Maximum_Draft	feet	Computed

Equations:

If Vessel_Draft_Forward > Vessel_Draft_Aft, then
Maximum_Draft = Dredging_Draft_Forward,

otherwise:

Maximum_Draft = Dredging_Draft_Aft

Step c: Determine the minimum dredging depth.

Parameters Used and Sources:

Parameter	Units	Data Source
Vessel_Draft_Forward	feet	Dredging_Data table
Water_Depth	feet	Dredging_Data table
Minimum_Dredging_Depth	feet	Computed

Equations:

Minimum_Dredging_Depth = 0.75 * (Vessel_Draft_Forward + Water_Depth)

Step d: Determine if material is being recovered.

Parameters Used and Sources:

Parameter	Units	Data Source
Pump_On_Port	Yes/No	Dredging_Data table
Pump_On_Stbd	Yes/No	Dredging_Data table
Pump_Material_Port	Yes/No	Dredging_Data table
Pump_Material_Stbd	Yes/No	Dredging_Data table
Pumping_Material	Yes/No	Computed

Equations:

Pumping_Material is true if:
Pump_On_Port = True, and
Pump_Material_Port = True, or if
Pump_On_Stbd = True, and
Pump_Material_Stbd = True

Step e: Determine if both dragheads are up.

Parameters Used and Sources:

Parameter	Units	Data Source
Port_Draghead_Depth Status	Ok, Lo, Hi, No	Dredging_Status table
Stbd_Draghead_Depth Status	Ok, Lo, Hi, No	Dredging_Status table
Water_Depth Status	Ok, Lo, Hi, No	Dredging_Status table
Draghead_Depth_Port	feet	Dredging_Data table
Draghead_Depth_Stbd	feet	Dredging_Data table
Maximum_Draft	feet	Computed

Minimum_Dredging_Depth	feet	Fixed
Dragheads_Up	Yes/No	Computed

Equations:

Dragheads_Up is true if:
Port_Draghead_Depth Status = Acceptable, and
Draghead_Depth_Port <= Maximum_Draft, and
Stbd_Draghead_Depth Status = Acceptable, and
Draghead_Depth_Stbd <= Maximum_Draft,

otherwise:

Water_Depth Status = Acceptable, and
Port_Draghead_Depth Status = Acceptable, and
Draghead_Depth_Port <= Minimum_Dredging_Depth, and
Stbd_Draghead_Depth Status = Acceptable, and
Draghead_Depth_Stbd <= Minimum_Dredging_Depth

Step f: Determine if vessel is turning.

Parameters Used and Sources:

Parameter	Units	Data Source
Gyro_Rate	degrees	Computed
Minimum_Turning_Rate	degrees	Fixed
Turning_Vessel	Yes/No	Computed

Equations:

Turning_Vessel is true if:
Gyro_Rate > Minimum_Turning_Rate

Step g: Determine the possible dredge state(s).

Determination of possible dredge states is summarized in the Dredge Status Computation Logic Table (Table 2). Each dredge state has a unique set of conditions. If the computations cannot determine a unique dredge state, the system attaches a "Downtime" dredge state designation to the data record.

Parameters Used and Sources:

Parameter	Units	Data Source
Pumping_Material	true/false	Computed
Pump_On_Port	true/false	Dredging_Data table
Pump_On_Stbd	true/false	Dredging_Data table
Ground_Speed_Status	Ok, Lo, Hi, No	Dredging_Status table
Vessel_Speed	knots	Dredging_Data table
Minimum_Dredging_Speed (0.1)	knots	Fixed
Course_Status	Ok, Lo, Hi, No	Dredging_Status table
Gyro_Rate	degrees	Computed
Minimum_Turning_Rate (.05)	degrees/sec	Fixed
Port_Draghead_Depth_Status	Ok, Lo, Hi, No	Dredging_Status table
Stbd_Draghead_Depth_Status	Ok, Lo, Hi, No	Dredging_Status table
Draghead_Depth_Port	feet	Dredging_Data table
Draghead_Depth_Stbd	feet	Dredging_Data table

Table 2
Dredge Status Computation Logic Table

Dredge State	Pump(s) On	Pumping Material	Drag-heads Up	Drag-heads <+ max. ship draft	Vessel Draft < draft empty	Vessel Draft > draft loaded	Gyro rate < min turn rate	Gyro rate >= min. turn rate	Vessel speed < min sailing speed	Vessel speed >= min. sailing speed	Vessel speed > min. dredging speed	Vessel speed > min. dumping speed	Hopper doors open
Sailing Empty	False	False	Yes		Yes			No		Yes			
Dredging Material	True	True									Yes		
Pumping (water)	True	False					Yes				Yes		
Turning		False	Yes	Yes				Yes	Yes				False
Sailing Full	False	False	Yes			Yes		No		Yes			
Dumping									Yes			Yes	True
Offloading													
Downtime													

Notes:
Pumps on when: computed in DSS.
Pumping material when: computed in DSS.
Dragheads up when: draghead depth <0.75* (vessel draft foreward + water depth under vessel hull).
Maximum ship draft: greater of the foreward or aft draft values.
Ship draft loaded: foreward = 17.0 ft; aft = 26.0 ft. (for *Essayons*).
Ship draft empty: foreward = 17.0 ft; aft = 24.5 ft. (for *Essayons*).
Minimum turning rate: .05 deg/sec.
Minimum sailing speed: 4.0 knots.
Minimum dredging speed: 0.1 knot.
Minimum dumping speed: 0.1 knot.

Maximum_Draft	feet	Computed
Minimum_Sailing_Speed (4.0)	knots	Fixed
Minimum_Dumping_Speed (0.1)	knots	Fixed
Dragheads_Up	true/false	Computed
Hopper_Door_Open	true/false	Dredging_Data table
State	none	Computed
Turning_Vessel	true/false	Computed
Forward_Draft_Status	Ok, Lo, Hi, No	Dredging_Status table
Aft_Draft_Status	Ok, Lo, Hi, No	Dredging_Status table
Vessel_Draft_Forward	feet	Dredging_Data table
Vessel_Draft_Aft	feet	Dredging_Data table
Forward_Draft_Loaded (24.0)	feet	Fixed; specific to Essayons
Aft_Draft_Loaded (26.0)	feet	Fixed; specific to Essayons
Forward_Draft_Empty (17.0)	feet	Fixed; specific to Essayons
Aft_Draft_Empty (24.5)	feet	Fixed; specific to Essayons

Equations:

Dredging_Material_State is true if:
Pumping_Material (results of Step 4) is True, and
Hopper_Door_Open is False

Pumping_State (no material recovery) is true if:
Pumping_Material (results of Step 4) is False, and
Pump_On_Port is True, or
Pump_On_Stbd is True

Pump_On_Stbd is False, and
Ground_Speed_Status = Acceptable, or
Vessel_Speed > Minimum_Dredging_Speed, and
Course_Status = Acceptable, or
Gyro_Rate < Minimum_Turning_Rate

Turning_State is true if:
Port_Draghead_Depth Status = Acceptable, and
Draghead_Depth_Port <= Maximum_Draft, and
Stbd_Draghead_Depth Status = Acceptable, and
Draghead_Depth_Stbd <= Maximum_Draft, and
Ground_Speed_Status = Acceptable, or
Vessel_Speed < Minimum_Sailing_Speed, and
Pump_Material_Port is False, and
Pump_Material_Stbd is False, and
Dragheads_Up, and
Pumping_Material is False, and
Hopper_Door_Open is False, and
Course_Status = Acceptable, or
Gyro_Rate >= Minimum_Turning_Rate, and
State = Pumping, Turning, or Down

Dumping_State is true if:
Hopper_Door_Open is True, and
Ground_Speed_Status = Acceptable, or
Vessel_Speed > Minimum_Dumping_Speed, and
Vessel_Speed < Minimum_Sailing_Speed, and
State = To-Dump, Dumping, or Down

Sailing_Full_State is true if:
Pump_On_Port is False, and
Pump_On_Stbd is False, and
Pumping_Material is False, and
Turning_Vessel is False, and
Dragheads_Up is True, and
Forward_Draft Status = Acceptable, and
Vessel_Draft_Forward > Forward_Draft_Loaded, and

Aft_Draft_Status = Acceptable, and
Vessel_Draft_Aft > Aft_Draft_Loaded, and
Ground_Speed_Status = Acceptable, or
Vessel_Speed >= Minimum_Sailing_Speed, and
State = Pumping, Turning, To_Dump, or Down

Sailing_Empty_State is true if:
Pump_On_Port is False, and
Pump_On_Stbd is False, and
Pumping_Material is False, and
Turning_Vessel is False, and
Dragheads_Up is True, and
Forward_Draft_Status = Acceptable, and
Vessel_Draft_Forward < Forward_Draft_Empty, and
Aft_Draft_Status = Acceptable, and
Vessel_Draft_Aft < Aft_Draft_Empty, and
Ground_Speed_Status = Acceptable, or
Vessel_Speed >= Minimum_Sailing_Speed, and
State = Dumping, To_Cut, or Down

Step h: Determine, rename, dredge state from possible states.

The selection of the actual dredge state to be assigned to a data record from the possible dredge state(s) computed under step g is performed using the following priority: dumping, pumping (material or water), to-cut, to-dump, turning, down. If dumping is computed within step g as a possible dredge state, then dumping is selected as the dredge state as it is given the highest selection priority. If dumping is not a possible dredge state, then the next dredge state in the priority list which is also included in the possible dredge states is assigned to the data record.

Dredging area (station)

A dredging area can be made up of 1 to 10 rectangular areas. For each dredging record where the dredge state is PUMPING, a test is made to determine if the current vessel position is within one of these rectangular areas. If so, a record is added to the ***LOAD-STATION*** table indicating that the dredge was dredging in that area during the current load. If the dredge is not pumping in an assigned dredging area, that too is recorded in the ***LOAD_STATION*** table.

Disposal area

A disposal area can be made up of 1 to 10 rectangular areas. For each dredging record where the dredge state is dumping, a test is made to determine if the current vessel position is within one of these rectangular areas. If so, a record is added to the ***LOAD_DISPOSAL_AREA*** table indicating that the dredge was dumping in that dredging area during the current load. If the dredge is not dumping in an assigned dredging area, that too is recorded in the ***LOAD_DISPOSAL_AREA*** table.

Time summations for reports

Trip, daily, and job-to-date reports contain summations of the time certain activities have occurred, such as the amount of time each dredge state has occurred within the reporting time period. These time summations are performed by the subroutines, initiated by ***RT Kernel***, which generate the reports. The time summations are a straight summation based on the times attached to each data record in the ***DREDGING_DATA*** table for a trip (load), day, or job-to-date.

6 System Interfaces

The Silent Inspector system contains four main interfaces between the systems components, and between the system and other types of hardware platforms and software. These interfaces and their purpose are:

Interface	Purpose
Sensors to DSS	Transfer of sensor data to the DSS component
DSS to SHIP	Transfer of data from DSS to SHIP database
SHIP to SHORE	Transfer of data from SHIP to SHORE components
Outside Access	Access to the SHIP or SHORE databases using other types of software

Overviews of each interface, and its specifications and format requirements, are provided below.

Sensors-to-DSS Interface

As each hopper dredge has a different suite of electronic sensors, the interface between the sensors, or a sensor data collection computer, and the DSS component will vary from dredge to dredge. It is anticipated that no two sensor-to-DSS interfaces may be exactly alike. What is important is that each DSS must take these varied inputs and create a standard, identical format data record for insertion into the SHIP database.

When the data are received by the DSS via RS232, the data should have already been converted into engineering units and data calibrations applied, so the data are considered final, true values. If other than RS232 data inputs are to be received, such as sensor voltages, the conversions of the sensor outputs to useable units and calibrated values may be programmed into the DSS.

The training and reference DSS was designed specifically for the Corps of Engineers hopper dredge *Essayons*, which already has a data logging computer aboard. The DSS was therefore designed to accept sensor data in RS232 format. So as to allow development and testing of the DSS and SHIP components, the DSS also accepts sensor data from pre-recorded data files of *Essayons* data. The sensor-to-DSS interface format was structured to match the format output by the *Essayon's* data logging computer. Each data record is 127 characters followed by a carriage return line-feed <crlf>. The specific sensor data in each incoming record, and its format, character length, and position within each 127-character record, are shown below.

Sensor Data (Parameter)	Units	Data Format	Character Length	Character Position
Flag	none	ASCII string	6	1 - 6
Date	yymmdd (local)	ASCII string	6	7-12
Time	seconds (local)	Floating point	5	13-17
RMS Error	feet	Floating point	4	18-21
X location	feet	Floating point	7	22-28
Y location	feet	Floating point	7	29-35
Forward draft	feet	Floating point	6	36-41
Aft draft	feet	Floating point	6	42-47
Tide elevation	feet	Floating point	5	48-52
Port drag arm velocity	feet/sec	Floating point	4	53-56
Port drag arm density	grams/liter	Floating point	4	57-60
Starboard drag arm velocity	feet/sec	Floating point	4	61-64
Starboard drag arm density	grams/liter	Floating point	4	65-68
Port gimbal depth	feet	Floating point	4	69-72
Starboard gimbal depth	feet	Floating point	4	73-76
Port draghead depth	feet	Floating point	4	77-80
Starboard draghead depth	feet	Floating point	4	81-84
Heading	degrees true	Floating point	3	85-87
Course	degrees true	Floating point	3	88-90
Water depth (below hull)	feet	Floating point	4	91-94
Speed (over ground)	knots	Floating point	4	95-98
Ship weight-hopper open	long tons	Floating point	6	99-104
Ship weight-hopper closed	long tons	Floating point	6	105-110
Ullage - meter no. 1	feet	Floating point	4	111-114
Ullage - meter no. 2	feet	Floating point	4	115-118
Ullage - meter no. 3	feet	Floating point	4	119-122
Ullage - meter no. 4	feet	Floating point	4	123-126
Hopper door open	true/false	ASCII string	1	127

Definitions of the data to be provided for each parameter are shown below. Specific value ranges or limitations specific to the *Essayons* data record are also listed. Note that for all parameters having a floating point data format, the position of the decimal place varies and is contained within the characters sent for each parameter.

Sensor Data (Parameter)	Definition
Date	Date in local time of the sensor measurements, formatted yymmdd.
Time	Seconds past midnight, from 0.000 to 86400, local time.
Root mean square (RMS) Error	RMS error of ship's position based on the X and Y range values used to calculate the position. Valid numbers range from 0.00 to 32.9, which is the maximum value tolerated in the *Essayon's* positioning system.
X location	X (easting) Lambert position of the dredge.
Y location	Y (northing) Lambert position of the dredge.

Forward and aft draft	Draft of vessel below waterline at the forward and aft sensor locations. On the *Essayons*, forward draft range = 10 to 30 ft, aft draft range = 20 to 30 ft.
Tide elevation	Tide height relative to mean lower low water.
Port & Stbd drag arm velocity	Velocity of water moving through the drag arms.
Port & Stbd drag arm density	Specific gravity of water/material mixture in the drag arms. Valid values are 1.0 to 2.0; values < 1.0 indicate no water in the drag arm.
Port & Stbd gimbal depth	Depth below water surface of the drag arm gimbals. Valid range for *Essayons* is 0.00 to 50.0.
Port & Stbd draghead depth	Depth below water surface of the low fixed point of each draghead. This value includes a correction for the draft and trim of the vessel, and is not depth below the keel.
Heading	Heading in degrees of the vessel as taken from the Gyro-compass. Values are from 0.00 to 359.
Course	Vessel course in degrees made good as computed from the vessel's navigation system data over 10-sec intervals.
Water depth	Depth below the keel at the location of the sensor. This sensor is not in place on the *Essayons*, and therefore no data are provided for this parameter.
Speed	Vessel speed over the ground as computed from the vessel's navigation system data over 10-sec intervals.
Ship weight	Weight of the vessel with the doors open and closed. Not currently included in the data record.
Ullage	Height of water in the hopper. This is computed as the distance between the water surface to the Ullage sensor, subtracted from the distance between the bottom of the hopper to the Ullage sensor. The *Essayons* has four Ullage sensors located in each corner of the hopper. A value for each sensor is included in the data record.
Hopper doors open	Status of the hopper doors as either open (true), all doors all the way closed (false), or undetermined (unknown). Any single hopper door open requires a door open status.

The training and reference DSS accepts data records as fast as the *Essayons* data logging computer sends such records. At present, data records are sent by the *Essayons* computer and received by the DSS every 10 sec. The required spacing for the receipt of sensor data by all DSS components is to be no more than 10-sec intervals.

DSS-to-SHIP Interface

After receiving input of a sensor data record, the DSS attaches three operator input and several DSS computed parameters to the data record and stores the complete outgoing record in a database format suitable for transfer to the

SHIP database. The additional parameters attached to each data record, and their source, are:

Parameter	**Units**	**Data Format**	**Source**
Contract ID	None	ASCII string	Operator input to DSS
Project name	None	ASCII string	Operator input to DSS
Dredge name	None	ASCII string	Operator input to DSS
Date_Time	local	Sybase datetime	Converted by DSS into Sybase datetime format from the date and time inputs
Average hopper level forward	feet	Floating point	Computed by DSS
Average hopper level aft	feet	Floating point	Computed by DSS
Average hopper level	feet	Floating point	Computed by DSS
Hopper volume	yards^3	Floating point	Computed by DSS
Port pump on	true/false	ASCII string	Computed by DSS
Starboard pump on	true/false	ASCII string	Computed by DSS
Port material recovery	true/false	ASCII string	Computed by DSS
Starboard material recovery	true/false	ASCII string	Computed by DSS
Pump out on	true/false	ASCII string	Computed by DSS
Status value for each parameter	ok, lo, hi, no	ASCII string	Computed by DSS

The definitions of the data for the parameters input by the operator or computed by the DSS are shown below. See Chapter 5 for details of specific computations.

Parameter	**Definition**
Contract ID	The unique contract number or identifier under which the dredging data were collected.
Project name	The name of the project under which the dredging data were collected.
Dredge name	The name of the dredge on which the dredging data were collected.
Date_Time	The date and time, in a Sybase Datetime format, at which the measurements in this record were taken. Sybase accepts the date and time in several different formats (e.g., 940930, 10:45:50.4AM). Refer to the Sybase Manuals for the acceptable formats.
Avg. hopper level forward	Average height of water or material in the forward end of the hopper as computed from the two forward ullage sensors.
Avg. hopper level aft	Average height of water or material in the aft end of the hopper as computed from the two aft ullage sensors.
Avg. hopper level	Average height of water or material in the hopper as computed from the computed forward and aft average hopper levels.
Hopper volume	The volume of water and material in the hopper as computed from the computed average hopper level and equations describing the hopper volume versus height in the hopper.
Port pump on	The port dredging pump is on (true), off (false), or undetermined (unknown) as computed from the Port Drag Arm Velocity value.
Starboard pump on	The starboard dredging pump is on (true), off (false), or undetermined (unknown) as computed from the Starboard Drag Arm Velocity value.

Port material recovery	The port dredging pump is pumping material (true), not pumping material (false), or undetermined if it is pumping material (unknown) as computed from the Port Drag Arm Density value.
Starboard material recovery	The starboard dredging pump is pumping material (true), not pumping material (false), or undetermined if it is pumping material (unknown) as computed from the Starboard Drag Arm Density value.
Pump-out on	The pump-out pump (for discharge of the hopper contents through a pipeline) is on (true), off (false), or undetermined (unknown) as computed from the Pump-Out Pump Velocity value. Note that this value is not provided in the *Essayons* 127-character data record input into the DSS.
Status of parameter values	A data quality flag assigned to all parameters having numerical values. The validity of each parameter value is determined by comparing the value against a table of acceptable ranges (see Chapter 3), and given the following data quality flags:

OK	-	measurement is acceptable
LO	-	measurement is out-of-range low
HI	-	measurement is out-of-range high
NO	-	measurement is missing

While each DSS will have different interface specifications between the sensors and the DSS, as well as different methods of handling the sensor data within the DSS to some extent, all DSS components must create identical completed data records stored within the DSS data structure and transfer these completed records into the SHIP database in identical fashion. All DSS-to-SHIP interfaces must therefore meet the following specifications.

Once the DSS has accepted a data record from the sensors, attached the three operator input parameter values, and computed the values for the additional parameters listed above, it stores the new completed record in a data structure within the DSS. Once in this data structure, the data record is now ready for transfer to the SHIP central database. If the SHIP component is not on-line, the DSS presently cannot transfer these data into the SHIP central database. These data can be written directly to a data file, as is the case with the training and reference DSS, and played back at a later time to insert into the SHIP central database. The data record within the DSS data structure is over-written as each new sensor record is obtained and completed by the DSS.

After a completed record has been inserted into the DSS database structure and the SHIP component is on-line, the DSS will insert the newly completed record into the ***DREDGING_DATA*** table within the SHIP component's central database. The DSS must do this by communicating directly with the SHIP database on a client-server basis. Communications between the DSS and SHIP database are performed in structured query language (SQL) format using Sybase Library calls and stored procedures. The SQL operation used to store the contents of the DSS database structure into the ***DREDGING_DATA*** table is the ***Insert*** operation. The ***Insert*** operation is presently invoked by calling the database interface package, written in ADA code, named ***Database***. The Sybase ***Insert*** operation for the ***DREDGING_DATA*** table is provided in that package. The ***Insert*** operation contained within the Sybase Open/ADA interface may also be used rather than the ***Database*** package. If the DSS code is written in C, the insert operation in the Sybase Open/C interface should be used. The stored procedure named ***INSERT_DREDGING_DATA*** may also be invoked directly from any language. The DSS does not interact with any other tables within the SHIP database, or with any other portion of the SHIP component.

Listed below are the parameters contained within the DSS data structure, the corresponding field name for the parameter in the ***DREDGING_DATA*** table, and the length, in bytes, of the field. The definition, units, and data format for each parameter remain the same as listed in tables above. All status parameters specify the validity of corresponding measurements (see Chapter 3). Note that values for ten parameters input into the DSS in the original 127-character sensor data record are not contained within the DSS data structure, nor inserted into the ***DREDGING_DATA*** table. These parameters are: port drag arm velocity, starboard drag arm velocity, port drag arm density, starboard drag arm density, port drag arm gimbal depth, starboard drag arm gimbal depth, and ullage meter numbers 1-4.

Parameter Name	***DREDGING_DATA*** **Field Name**	**Field Length**
Contract ID (number)	CONTRACT_ID	32
Project name	PROJECT	32
Dredge name	DREDGE	32
Date & time	DATE_TIME	8
X location	VESSEL_X	8
X location data status	VESSEL_X_STATUS	2
Y location	VESSEL_Y	8
Y location data status	VESSEL_Y_STATUS	2
Forward draft	VESSEL_DRAFT_FORWARD	8
Forward draft data status	VESSEL_DRAFT_FORWARD_STATUS	2
Aft draft	VESSEL_DRAFT_AFT	8
Aft draft data status	VESSEL_DRAFT_AFT_STATUS	2
Speed (over ground)	VESSEL_SPEED	8
Speed data status	VESSEL_SPEED_STATUS	2
Heading	VESSEL_HEADING	8
Heading data status	VESSEL_HEADING_STATUS	2
Course	VESSEL_COURSE	8
Course data status	VESSEL_COURSE_STATUS	2
Port draghead depth	DRAGHEAD_DEPTH_PORT	8
Port draghead depth data status	DRAGHEAD_DEPTH_PORT_STATUS	2
Stbd draghead depth	DRAGHEAD_DEPTH_STBD	8
Stbd draghead depth data status	DRAGHEAD_DEPTH_STBD_STATUS	2
Avg. hopper level forward	HOPPER_LEVEL_FORWARD	8
Avg. hopper level frwd data status	HOPPER_LEVEL_FORWARD_STATUS	2
Avg. hopper level aft	HOPPER_LEVEL_AFT	8
Avg. hopper level aft data status	HOPPER_LEVEL_AFT_STATUS	2
Avg. hopper level	HOPPER_ULLAGE	8
Avg. hopper level data status	HOPPER_ULLAGE_STATUS	2
Hopper volume	HOPPER_VOLUME	8
Hopper volume data status	HOPPER_VOLUME_STATUS	2
Water depth	WATER_DEPTH	8
Water depth data status	WATER_DEPTH_STATUS	2
Tide elevation	TIDE	8
Tide elevation data status	TIDE_STATUS	2
Hopper door status	HOPPER_DOOR_OPEN	7
Port pump on	PUMP_ON_PORT	7
Starboard pump on	PUMP_ON_STBD	7
Port material recovery	PUMP_MATERIAL_PORT	7

Starboard material recovery	PUMP_MATERIAL_STBD	7
Pump-out on	PUMP_OUT_ON	7

Each data record transmitted from the DSS to SHIP must represent sensor data collected at a maximum interval of 10 sec apart. In the case of the training and reference DSS, recorded data files may also be transmitted to the SHIP database. In this case, the user may redefine the time associated with each data record after the first data record by setting a time interval to be added to the first record's time value, as described earlier in Chapter 3. The DSS then transmits all the recorded data records within a file at the maximum possible computer communications speed. This allows for faster playback and testing of the system than is possible otherwise. This feature may not necessarily be provided on any production DSS component.

SHIP-to-SHORE Interface

Currently the SHIP-to-SHORE interface is performed using the Archive module. This module copies the data from six SHIP database tables into Unix archive files, which are then transferred to a storage medium (tape) at the user's direction. These files may then be loaded into any DOSIS database residing within a SHIP or SHORE component. The format for these archive files is described in the "Archive Files and Procedures" section of this manual.

Outside Access Interface

No specific interface has been developed to allow other software and hardware platforms to interact directly with the Silent Inspector components and software. There is, however, an interface built directly into the Sybase Database program that allows other software to access the DOSIS database on the SHIP or SHORE component, provided that the proper user ID and password are provided.

The Sybase database contains the Open Database Connectivity (ODBC) driver. External software programs containing this driver can be linked and data from the DOSIS database can be either inserted or copied. Insertion of data is permitted only into selected database tables where user input is required. Data may be copied from all tables. Tables containing measured or computed values are read only, and cannot be altered by external software programs.

7 SHIP Software Modules

The SHIP component contains the database server executing the relational DBMS. SHIP also executes user-initiated application software that operates as clients of the DBMS, accessing information in the database to perform database maintenance, real-time data monitoring, reporting, and graphical data displays. The SHIP applications software contains several major subsystems, or modules. Each user-invoked module is either a stand-alone operational component, or is composed of multiple operational components. Each module is user initiated through graphical user interface screens and icons. The user-invoked modules presently contained within the SHIP system and their function are:

Module	Function
Setup	User entry of project and dredge information.
Monitoring	User viewing of incoming data.
Downtime	User entry of downtime causes and comments (identification of downtime events is performed automatically within ***RT Kernel***, which is not user-invoked).
Reports	User-initiated display and printing of pre-defined reports.
Plotting	User viewing of data within the database.
Backup	Automatic backup of selected database tables and data; user-initiated restoration of the database from the backup files.
Archive	Combination automatic and user-initiated archiving of selected database tables and data for transfer to SHORE component and permanent record keeping.

These modules and their components are described briefly below.

Setup

Information related to a dredging project, including the dredge name and characteristics, crew, designated dredging and disposal areas, and landmarks related to the local marine waters can be entered into the system by the user. This information is entered into the system through the system's Setup module.

The information is entered by the user into one of six SHIP database tables. The table names and the information which can be entered into each are listed below. Characteristics of each data element within these tables are more completely described in Table 1. Information the user can enter into each table is listed below.

CREWMEMBER Table:
- First name
- Last name
- Rank
- ID (Initials)

DISPOSAL_AREA Table:
- Disposal area name
- Four x, y coordinates forming a rectangle to represent the disposal area

DISTRICT Table:
- Corps of Engineers District names
- Abbreviations for the District names

DREDGE Table:
- Dredge name
- Dredge owner name
- Dredge type
- Year built
- Vessel beam
- Vessel length
- Vessel dredge pumps horsepower
- Vessel displacement: empty and full
- Vessel maximum speed: empty and full
- Vessel draft: fore and aft when empty and loaded
- Distance to draft sensors along vessel hull
- Hopper maximum capacity
- Distance in hopper to ullage sensors

LANDMARK Table:
- Landmark name
- x, y coordinates of landmark location

PROJECT Table:
- Project name
- Contract number (ID)
- Contractor name
- Corps of Engineers District name
- Project type: new work or maintenance
- Anticipated start/finish dates

STATION (Dredging areas) Table:
- Dredging area name
- x, y coordinates forming rectangle to represent the dredging area

The SETUP module is activated by clicking on the SETUP icon, and then clicking on the icon for the type of data the user wishes to enter. Optionally, the user may use a pull-down menu to access the components within the SET-UP module. In either case, the user is supplied with a screen listing entry boxes for all the information which may be entered or changed related to the database table selected (type of information being input). It is intended that information may be entered via the SETUP module either directly on a SHIP system computer or into a portable laptop computer. Information can be entered into the SHIP database tables via connectivity options provided by the ODBC capabilities of the Sybase DBMS.

Monitoring

The **DATA MONITORING** module provides a real-time display of the data being stored by the DSS in the SHIP database. It accomplishes this by performing the following five major functions:

a. Displays the most current set of data values stored by the DSS in the *DREDGING_DATA* table.

b. Computes and displays the load number associated with the displayed data values.

c. Computes and displays the current TDM value associated with the displayed data values.

d. Plots the computed TDM values over a short time period to allow the user to see the trend of the vessel's loading.

e. Computes and displays the current dredge state associated with the displayed data values.

The application which accomplishes the above actions is written in Ada code and is started by clicking on the **DATA MONITORING** icon on the SHIP main menu screen.

Downtime

The **DOWNTIME** module permits the user to display information about the dredge downtime periods which have been identified and recorded as such in the database by the system, and to annotate any downtime period with a cause and a comment, both of which are then stored with the downtime record within the database. The operator may select to view all downtime events recorded in the database, or only those for which no cause or comment has already been entered. Information entered by the user is entered into the *DOWNTIME* table within the database. The application to accomplish the user access and selection of a downtime event, and entry of a cause and comment, is written in ADA code, and is started by clicking on the **DOWNTIME** icon on the SHIP main menu screen.

Reports

The **REPORT** module provides the user the ability to display and print the following pre-defined reports: trip report, daily report, and job report. Each report is displayed and printed by a separate software component addressed through the overall **REPORT** module written in Ada code. The reports contain the following:

Trip Report	Lists the start time and associated data values for all dredge state changes that occur during a trip (load). The report contains one entry for each dredge state change.
Daily Report	Lists data values for each trip started during a day, along with summations of relevant values and duration of each dredge state over the trips listed in the report.
Job Report	Lists summations of relevant values and duration of each dredge state from the beginning of a project to the date of the report. This component also updates the database table ***PROJECT SUMMARY*** with this information at the user's option.

The **REPORT** module allows the user, by clicking on the appropriate icon, to review a list of the trip or daily reports contained within the database for any project identified by any contract ID and dredge name. The user may select the report of interest from the displayed list by clicking on the trip number or day of interest. The report will then be displayed and optionally printed. The user cannot change or edit the report.

The job report is always regenerated from the data within the database whenever the user invokes this component of the **REPORT** module. The newly generated job totals will first be displayed and, at the option of the user, printed and/or entered as updated information within the database.

Plotting

The **PLOTTING** module permits the user to interactively select data within the database and generate time series plots of the selected data. This module uses general purpose plotting software developed specifically for the Silent Inspector system. The **PLOTTING** module's overall controlling software is in ADA code. This code is activated by clicking on the **PLOTTING** icon on the SHIP main menu screen.

Currently the user can select up to four data elements to be plotted at the same time on the Y axis versus time on the X axis. The values for the data elements selected can be obtained from the database by selecting a trip number, in which case all values for the trip (for the data elements selected) will be plotted, or by time interval, in which case only those values within the specified time interval will be plotted. All data elements to be plotted together must have the same units so that they may be plotted on the same plot. Time along the X axis is plotted at a specified scale; therefore, if the data being plotted span a longer time than can be displayed on the screen, a horizontal slider bar is provided to allow the user to scroll through the data.

Backup and Archive

Backup and archive are two separate modules which operate in a similar manner. The function of the backup module is to automatically copy database data to backup files which would enable the SHIP database to resume operating after a catastrophic loss. While creation of the backup files is automatic, restoring the SHIP database using the backup files must be user initiated.

The function of the archive module is to remove old data from the SHIP database for transmission to SHORE components and permanent record keeping. The archive module automatically transfers data to Unix files, then the user must initiate transfer of the Unix files to another medium (disk, tape, etc.) for transfer, storage, and loading into a SHORE component.

The backup and archive modules copy data contained in six user tables within the SHIP database to external files. These six tables are:

- DOWNTIME
- DREDGING_DATA
- LOAD
- LOAD_DISPOSAL_AREA
- LOAD_STATION
- STATE

The backup and archiving modules are currently not accessible and cannot be initiated from the graphical user interface screens specific to the Silent Inspector system. These modules are different from most of the other user-invoked modules in that they are not controlled by ADA code, but instead are currently launched using Unix shell scripts entered on the main Unix user interface screen. All scripts are written using Unix c shell (csh) code. Because these Unix scripts are not launched by clicking on icons to activate ADA code, they are described and listed below, as the user must enter them to activate certain parts of the backup and archive processes.

Unix environment variables

Certain Unix environment variables must be set to permit the backup/ recovery and the archive/restore facilities to work correctly. They may be set for an individual user by adding their definitions to the user's .login file for the c shell or the .profile file for the Bourne shell in the user's home directory. They may be set for all users by adding their definitions to the cshrc file for the c shell or the profile file for the Bourne shell in the /etc directory. The syntax for setting an environment variable is shown below for both shells.

c shell	setenv variable-name value Example: setenv DOSIS_ARCHIVE /usr/dosis/archive
Bourne shell	variable-name=value export variable-name Example: DOSIS_ARCHIVE=/usr/dosis/archive export DOSIS_ARCHIVE

The following list contains the environment variables that must be defined for the backup/recovery and archive/restore facilities. Note that the variable names are in upper case. A brief description is included for each variable.

Variable	Description
DOSIS_ARCHIVE	The full path name of the directory used by the archive and restore facility to store files. During an archive operation, the files written from the database are stored in this directory prior to being written to tape. During a restore operation, the files read from tape are stored in this directory prior to being loaded into the database.
DOSIS_BACKUP	The full path name of the directory used by the backup and recovery facility to store files. The database dump file and the transaction log files are stored in this directory.
DOSIS_COMMAND_DIR	The full path name of the directory that contains the command files (e.g., dosis_restore, dosis_clean, etc.) used by the operator to initiate recovery, archive and restore operations.
DOSIS_DUMP_DEVICE	The device name of the Sybase dump device on which the database backup and transaction log files are written. Refer to the description of the sp_addumpdevice command in the Sybase Manuals.
DOSIS_TAPE	Default tape device (e.g., /dev/rmt0 for QIC tape device) on which archive files are written during an archive operation or read during a restore operation.
SA_PASSWORD	The Sybase password to be used when the recovery, archive, and restore commands signon to isql.

Although all variables do not have functionality in all scripts, all variables must be set when running any script. This is because a common script, check_vars, is called from all the other scripts to verify that environment variables have been set. Any script that automatically runs one of the DOSIS backup and archive scripts will have to set these environment variables appropriately.

The check_vars script is called from all the other backup or archive scripts to verify that all required environment variables have been set. Specifically, it performs the following checks:

a. SA_PASSWORD environment variable is set, and is, in fact, the Sybase password for the user "sa."

b. DOSIS_ARCHIVE environment variable is set and designates a directory.

c. DOSIS_BACKUP environment variable is set and designates a directory.

d. DOSIS_COMMAND_DIR environment variable is set and designates a directory.

e. DOSIS_DUMP_DEVICE environment variable is set and designates a dump device on the Sybase SQL server.

The check_vars script also sets several shell programming variables for the convenience of other scripts.

Backup files and procedures

There are two types of backup files: database backup and transaction log backup files. The file created by a database backup includes the entire contents of the six SHIP database tables listed earlier in this section, whereas the file created by a transaction log backup includes only data added to those tables since the last transaction log backup. The backup module performs a database backup once a week, and a transaction log backup once a day, although these frequencies may be adjusted by a systems engineer. One previous cycle of database and transaction log backup files are maintained, and older backup files are automatically deleted. Database backup files are named: yyyymmddX.dbd, where yyyymmdd represents the computer's numeric encoding for the date the backup was performed and X is a character suffix (a to z) used to sequence database backups made on the same day. Transaction log backup files are named: yyyymmddX.tlb, where the variables are the same as for the database backup name.

If a SHIP database loss occurred, the user must initiate restoration of the SHIP database using the backup files. This will not occur automatically. The Unix script to perform this action is entered into a Unix window accessed by clicking on the Unix main screen, and then clicking on K-Shell within the Unix menu. The Unix backup scripts and their purpose are:

Unix Script	Purpose
dosis_dumpdb	Dumps the entire contents of the six SHIP database tables to Unix backup files once per week.
dosis_dumptran	Dumps the data acquired within the six SHIP database tables since the last transaction log dump to Unix backup files once per day.
dosis_restore	Restores the contents of the six SHIP database tables using the database backup and transaction log Unix backup files.

The Unix scripts dosis_dumpdb and dosis_dumptran are automatically activated, while the dosis_restore script must be user entered. The specific actions which occur with each script are listed below.

dosis_dumpdb Script

The dosis_dumpdb script performs a Sybase database dump of the DOSIS database to begin a new backup cycle. In addition, it automatically deletes all old database and transaction log dumps, except for the most recent. The following specific operations are performed:

a. Remove the files for all old database dumps, except for the most recent.

b. Remove all transaction log dumps corresponding to each database dump removed.

c. Determine the correct name for the new database dump. (Dump files are named with the date, as yyyymmdd, followed by an alphabetic suffix (a, b, c, etc.) to sequentially indicate dumps on that date. Provision is made for several backup dumps occurring on the same day, since various archive operations can cause data backup dumps to occur.

d. Ensure that the 'select into' option (set for various archive operations, as noted herein) is turned off for the DOSIS database.

e. Truncate the transaction log in the DOSIS database.

f. Dump the DOSIS database and rename the dump file as required by the DOSIS backup naming conventions. The dump command used on the Sybase SQL server causes the dump to always be sent to the same file; the file must be renamed to save it in accordance with the DOSIS dump naming conventions.

Dosis_dumpdb is automated to run once a week; however, this frequency can be altered if desired. As noted elsewhere, database dumps are also taken as part of various archive operations, and the operator can also explicitly initiate a database dump at any time.

dosis_dumptran Script

The dosis_dumptran script performs a Sybase transaction log dump of the DOSIS database to save the latest incremental dump within a backup cycle. The following specific operations are performed.

1. Determine the correct name for the new transaction log dump. (Dump files are named with the date, as yyyymmdd, followed by an alphabetic suffix (a, b, c, etc.) to sequentially indicate dumps on that date. Provision is made for several backup dumps occurring on the same day, since various archive operations cause backup dumps to be recorded.

2. Dump the DOSIS transaction log and rename the dump file as required by the DOSIS backup naming conventions. The dump command used on the Sybase SQL server causes the dump to always be sent to the same file; the file must be renamed to save it in accordance with the DOSIS dump naming conventions.

It is planned that dosis_dumptran will be automated to run once a day; however, this frequency can be altered if desired. As noted elsewhere, transaction log dumps are also taken as part of various archive operations, and the operator can explicitly initiate a database dump.

dosis_restore Script

The dosis_restore script loads the latest database dump, followed by subsequent transaction log dumps, to restore the DOSIS database after it has been lost due to some catastrophic circumstance. The dosis_restore script will be run only upon explicit operator request. The following specific actions are performed:

a. Determine the name of the latest database dump (or give an error if no database dumps are available.)

b. Load the latest database dump.

c. Load, in sequence, all transaction log dumps taken subsequent to that database dump.

Once the data recovery process has been completed, the Silent Inspector system must be restarted by re-entering a user ID and password.

Archive files and procedures

Data from the six database tables listed earlier are automatically archived to Unix files once a month. This frequency may be changed by a systems engineer. Users may also initiate the archiving process. The archiving process also deletes the data from the six SHIP database files after it has copied this data to the Unix files. The Unix archive files are named yyyymmddXXX.ad, where yyyymmdd designates the date of archiving, and XXX designates the database table name. The XXX abbreviations for the table names are:

SHIP Database Table Name	XXX Abbreviation
DOWNTIME	DWN
DREDGING_DATA	DDA
LOAD_DISPOSAL_AREA	LDA
LOAD_STATION	LST
LOAD_TABLE	LDT
STATE	STA

Once the Unix archive files have been created, the user must initiate the transfer or copying of the files to other medium, such as disk or tape, deleting the Unix files from the hard drive, and loading the files into a separate SHIP or SHORE unit. Unix script to perform these actions, as well as initiate the archive process, are entered into a Unix window accessed by clicking on the Unix main screen, and then clicking on K-Shell within the Unix menu. The Unix scripts and their purposes are:

Unix Script	Purpose
dosis_archive	Forces an archive of the six database tables to Unix files, and deletes these data from the SHIP database tables.
dosis_tape	Copies the Unix archive files for a given date to a tape.
dosis_clean	Deletes Unix archive files selected by the user from the system hard drive.
dosis_load	Loads archived files into a SHIP or SHORE database.

The following specific actions occur for each script.

dosis_archive Script

The dosis_archive script writes data for all completed loads to six archive files (corresponding to the six SHIP database tables), then deletes all data within the SHIP tables which were archived. The following specific operations are performed.

1. Verify that an archive has not already been performed on the current day (Archive files are named to include the date and should not be overwritten since archived data are deleted from the database.)

2. Dump the Sybase transaction log. This is done to enable recovery to the time of the archive if something goes wrong with the deletion of data during the archive process.

3. Enable "select into" in the DOSIS database. This is done to allow the archived data to be subset from the database via "select into" (the most expedient method).

4. Bulk copy data for the six dynamic tables to the archive files. Various checks are made to ensure that all data that are supposed to be archived have been successfully bulk copied.

5. Delete data that had been archived from the six dynamic tables. The deletions are done after the archive is complete, so that no data will be deleted if any archive operation fails.

6. Dump the DOSIS database (via script dosis_dumpdb, which also resets the "select into" option). This is done because the 'select into' operation disables future transaction log dumps, so it is necessary to dump the entire database to begin a new backup cycle.

dosis_tape Script

The dosis_tape script copies Unix archive files (created by the dosis_archive script) for a given date to a tape device and will run only upon explicit operator request. The archive files are not deleted after the tape is created in case it is desired to make several tapes of the same archive. The dosis_clean script is provided to remove all archive files for a given date. The following specific actions are performed:

a. Decide whether a date has been specified as an argument. If no date is specified, then the date of the oldest archive is used. If the command line contains two arguments, the first is the date (in yyyymmdd format), and the second is the tape device to use. If the command line contains only one argument, the single argument could be either the date or the tape device; dosis_tape assumes that it is a date if it begins with 19 or 20; otherwise, the single argument is taken to be a device name. If the command line contains no arguments, then neither the date nor the tape device has been explicitly specified.

b. Decide whether or not a tape device has been specified as an argument. If no tape device is specified, the value of the environment variable DOSIS_TAPE is used for the name of the tape device.

c. Verify that the tape device specified is a valid device.

d. Verify that all archive files are present for the date to be used.

e. Create a saved data set on the target tape device containing the archive files from the appropriate date for the six dynamic tables.

dosis_clean Script

The dosis_clean script deletes all archive files for a specified date or for the oldest date stored, if no date is specified as a parameter. This feature only runs by explicit operator action. The following specific operations are performed.

a. Determine whether or not a date has been given as a parameter.

b. Use the date specified or the oldest date on an archive file to delete all archive files for that date.

dosis_load Script

The dosis_load script loads a set of archived data into a DOSIS database. The dosis_load script will typically be run only at shore sites to load archived data received from various dredges. The following specific actions are performed.

a. Determine the name of the tape drive from which the archived data are to be loaded. The tape drive may be specified by command line argument or by the value of the environment variable DOSIS_TAPE.

b. Verify that the tape drive specified is a valid device.

c. Copy the files from the tape to the $DOSIS_ARCHIVE directory.

d. Verify that the correct number of files have been read from the tape.

e. Determine the date of the archive from the names of the files and verify that all files came from the same date.

f. Verify that files corresponding to all dynamic tables saved in the archive have been read from the tape. (There is one file for each table, since the file is created by bulk copy. The name of the table is encoded in the name of the file.)

g. Dump the Sybase transaction log. This is done to enable recovery to the time of the load if a partial load causes inconsistent data to be entered into the database.

h. Enable "select into" in the DOSIS database. This is done to allow the archived data to be loaded via bulk copy (the most expedient method) and to allow simple script creation of a table used to verify that duplicate data are not being loaded.

i. Verify that data being loaded do not duplicate data already present in the database. This is done by copying the archived LOAD_TABLE into a table specifically created for this purpose, then seeing if the most current archived load already appears in the database.

j. Bulk copy data for the six dynamic tables into the database from the archive files.

k. Delete the archive files on disk.

l. Dump the DOSIS database (via script dosis_dumpdb which also resets the 'select into' option). This is done because the select into operation disables future transaction log dumps; so it is necessary to dump the entire database to begin a new backup cycle.

Appendix A
Stored Procedures

The stored procedures used with the Silent Inspector software are listed below in bold, along with their purpose and coding. These procedures are stored in a file that can be submitted to Sybase using interactive structured query language (isql). When this file is submitted to Sybase via isql, any existing stored procedure having the same name is deleted and the new procedure is created. The file name is procscomp.sql, and it can be submitted to isql as follows:

```
isql < procscomp.sql
```

The stored procedures for the Silent Inspector system are:

INSERT_DOWNTIME_DATA

Purpose: Insert a downtime event in the DOWNTIME table.

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'INSERT_DOWNTIME_DATA' )
  drop proc INSERT_DOWNTIME_DATA
go
create proc INSERT_DOWNTIME_DATA
  ( @CONTRACT_ID      CONTRACT_IDENTIFIER,
    @PROJECT          PROJECT_NAME,
    @DREDGE           DREDGE_NAME,
    @START_DATE_TIME datetime,
    @END_DATE_TIME   datetime,
    @LOAD_NO          LOAD_NUMBER,
    @CAUSE            DOWNTIME_CAUSE,
    @COMMENT          FREE_FORM_TEXT ) as
Insert into DOWNTIME
values
( @CONTRACT_ID,
  @PROJECT,
  @DREDGE,
  @START_DATE_TIME,
  @END_DATE_TIME,
  @LOAD_NO,
  @CAUSE,
  @COMMENT )
go
```

INSERT_DREDGING_DATA

Purpose: Insert dredging data in the DREDGING_DATA table.

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'INSERT_DREDGING_DATA' )
 drop proc INSERT_DREDGING_DATA
go
create proc INSERT_DREDGING_DATA
 ( @CONTRACT_ID                 CONTRACT_IDENTIFIER,
   @PROJECT                     PROJECT_NAME,
   @DREDGE                      DREDGE_NAME,
   @DATE_TIME                   datetime,
   @VESSEL_X                    COORDINATE_TYPE,
   @VESSEL_X_STATUS             STATUS,
   @VESSEL_Y                    COORDINATE_TYPE,
   @VESSEL_Y_STATUS             STATUS,
   @VESSEL_DRAFT_FORWARD        FEET_TYPE,
   @VESSEL_DRAFT_FORWARD_STATUS STATUS,
   @VESSEL_DRAFT_AFT            FEET_TYPE,
   @VESSEL_DRAFT_AFT_STATUS     STATUS,
   @VESSEL_SPEED                KNOTS_TYPE,
   @VESSEL_SPEED_STATUS         STATUS,
   @VESSEL_HEADING              DEGREES_TYPE,
   @VESSEL_HEADING_STATUS       STATUS,
   @VESSEL_COURSE               DEGREES_TYPE,
   @VESSEL_COURSE_STATUS        STATUS,
   @DRAGHEAD_DEPTH_PORT         FEET_TYPE,
   @DRAGHEAD_DEPTH_PORT_STATUS  STATUS,
   @DRAGHEAD_DEPTH_STBD         FEET_TYPE,
   @DRAGHEAD_DEPTH_STBD_STATUS  STATUS,
   @HOPPER_LEVEL_FORWARD        FEET_TYPE,
   @HOPPER_LEVEL_FORWARD_STATUS STATUS,
   @HOPPER_LEVEL_AFT            FEET_TYPE          ,
   @HOPPER_LEVEL_AFT_STATUS     STATUS             ,
   @HOPPER_VOLUME               CUBIC_YARDS_TYPE   ,
   @HOPPER_VOLUME_STATUS        STATUS             ,
   @HOPPER_ULLAGE               FEET_TYPE          ,
   @HOPPER_ULLAGE_STATUS        STATUS             ,
   @WATER_DEPTH                 FEET_TYPE          ,
   @WATER_DEPTH_STATUS          STATUS             ,
   @TIDE                        FEET_TYPE          ,
   @TIDE_STATUS                 STATUS             ,
   @HOPPER_DOOR_OPEN            BOOLEAN_TYPE       ,
   @PUMP_ON_PORT                BOOLEAN_TYPE       ,
   @PUMP_ON_STBD                BOOLEAN_TYPE       ,
   @PUMP_MATERIAL_PORT          BOOLEAN_TYPE       ,
   @PUMP_MATERIAL_STBD          BOOLEAN_TYPE       ,
   @PUMP_OUT_ON                 BOOLEAN_TYPE       ,
   @LOAD_NO                     LOAD_NUMBER        ,
   @STATE                       DREDGE_STATE       ,
   @TDM                         LONG_TONS_TYPE     ) as
insert into DREDGING_DATA
values
 ( @CONTRACT_ID          ,
   @PROJECT              ,
   @DREDGE               ,
   @DATE_TIME            ,
   @VESSEL_X             ,
   @VESSEL_X_STATUS      ,
   @VESSEL_Y             ,
   @VESSEL_Y_STATUS      ,
   @VESSEL_DRAFT_FORWARD        ,
   @VESSEL_DRAFT_FORWARD_STATUS,
   @VESSEL_DRAFT_AFT             ,
   @VESSEL_DRAFT_AFT_STATUS    ,
   @VESSEL_SPEED                ,
   @VESSEL_SPEED_STATUS         ,
   @VESSEL_HEADING              ,
   @VESSEL_HEADING_STATUS       ,
   @VESSEL_COURSE               ,
   @VESSEL_COURSE_STATUS        ,
   @DRAGHEAD_DEPTH_PORT         ,
   @DRAGHEAD_DEPTH_PORT_STATUS ,
   @DRAGHEAD_DEPTH_STBD         ,
   @DRAGHEAD_DEPTH_STBD_STATUS ,
   @HOPPER_LEVEL_FORWARD        ,
   @HOPPER_LEVEL_FORWARD_STATUS,
   @HOPPER_LEVEL_AFT            ,
   @HOPPER_LEVEL_AFT_STATUS    ,
   @HOPPER_VOLUME              ,
   @HOPPER_VOLUME_STATUS        ,
   @HOPPER_ULLAGE               ,
   @HOPPER_ULLAGE_STATUS        ,
   @WATER_DEPTH                 ,
   @WATER_DEPTH_STATUS          ,
   @TIDE                        ,
   @TIDE_STATUS                 ,
   @HOPPER_DOOR_OPEN            ,
   @PUMP_ON_PORT                ,
   @PUMP_ON_STBD                ,
   @PUMP_MATERIAL_PORT          ,
   @PUMP_MATERIAL_STBD          ,
   @PUMP_OUT_ON                 ,
   @LOAD_NO                     ,
   @STATE                       ,
   @TDM                         )
go
```

INSERT_LOAD_DATA

Purpose: Insert summary data for a load in the LOAD table.

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'INSERT_LOAD_DATA' )
  drop proc INSERT_LOAD_DATA
go
create proc INSERT_LOAD_DATA
( @CONTRACT_ID        CONTRACT_IDENTIFIER,
  @PROJECT            PROJECT_NAME,
  @DREDGE             DREDGE_NAME,
  @LOAD_NO            LOAD_NUMBER,
  @DATE               datetime,
  @PUMPING_TIME       MINUTES_TYPE,
  @TURNING_TIME       MINUTES_TYPE,
  @SAILING_FULL_TIME  MINUTES_TYPE,
  @DUMPING_TIME       MINUTES_TYPE,
  @SAILING_EMPTY_TIME MINUTES_TYPE,
  @DOWN_TIME          MINUTES_TYPE,
  @TOTAL_TIME         MINUTES_TYPE,
  @TDM                LONG_TONS_TYPE,
  @COMMENT            FREE_FORM_TEXT ) as
insert into LOAD_TABLE
```

```
values
( @CONTRACT_ID,
  @PROJECT,
  @DREDGE,
  @LOAD_NO,
  @DATE,
  @PUMPING_TIME,
  @TURNING_TIME,
  @SAILING_FULL_TIME,
  @DUMPING_TIME,
  @SAILING_EMPTY_TIME,
  @DOWN_TIME,
  @TOTAL_TIME,
  @TDM,
  @COMMENT )
go
```

INSERT_LOAD_DISPOSAL_AREA

Purpose: Insert a dump event for a rectangular disposal area in the LOAD_DISPOSAL_AREA table.

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'INSERT_LOAD_DISPOSAL_AREA' )
  drop proc INSERT_LOAD_DISPOSAL_AREA
go
create proc INSERT_LOAD_DISPOSAL_AREA
  ( @CONTRACT_ID               CONTRACT_IDENTIFIER,
    @PROJECT                   PROJECT_NAME,
    @DREDGE                    DREDGE_NAME,
    @LOAD_NO                   LOAD_NUMBER,
    @DISPOSAL_AREA             AREA_NAME,
    @DISPOSAL_START_DATE_TIME datetime ) as
insert into LOAD_DISPOSAL_AREA
values
( @CONTRACT_ID,
  @PROJECT,
  @DREDGE,
  @LOAD_NO,
  @DISPOSAL_AREA,
  @DISPOSAL_START_DATE_TIME )
go
```

INSERT_LOAD_STATION

Purpose: Insert a dredging event for a station (rectagular dredging area) in the LOAD_STATION table.

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'INSERT_LOAD_STATION' )
  drop proc INSERT_LOAD_STATION
go
create proc INSERT_LOAD_STATION
  ( @CONTRACT_ID               CONTRACT_IDENTIFIER,
    @PROJECT                   PROJECT_NAME,
    @DREDGE                    DREDGE_NAME,
    @LOAD_NO                   LOAD_NUMBER,
    @STATION_AREA              AREA_NAME,
    @STATION_START_DATE_TIME datetime ) as
insert into LOAD_STATION
```

```
values
( @CONTRACT_ID,
  @PROJECT,
  @DREDGE,
  @LOAD_NO,
  @STATION_AREA,
  @STATION_START_DATE_TIME )
go
```

INSERT_PROJECT_DREDGE

Purpose: *Insert a dredge assigned to a project in the PROJECT_DREDGE table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'INSERT_PROJECT_DREDGE' )
  drop proc INSERT_PROJECT_DREDGE
go
create proc INSERT_PROJECT_DREDGE
( @CONTRACT_ID              CONTRACT_IDENTIFIER,
  @PROJECT                  PROJECT_NAME,
  @DREDGE                   DREDGE_NAME,
  @CAPTAIN                  CREW_ID,
  @SPEED_TOLERANCE          KNOTS_TYPE,
  @DEPTH_TOLERANCE          FEET_TYPE,
  @DRAFT_LIMIT              FEET_TYPE,
  @DRAG_LIMIT               FEET_TYPE,
  @TRIM_LIMIT               FEET_TYPE,
  @TURN_LIMIT               DEGREES_PER_MINUTE_TYPE,
  @DRY_MATERIAL_MASS_DENSITY GRAMS_PER_LITER_TYPE,
  @WATER_MASS_DENSITY       GRAMS_PER_LITER_TYPE,
  @DREDGE_AIDS              FREE_FORM_TEXT,
  @COMMENT                  FREE_FORM_TEXT ) as
insert into PROJECT_DREDGE
values
( @CONTRACT_ID,
  @PROJECT,
  @DREDGE,
  @CAPTAIN,
  @SPEED_TOLERANCE,
  @DEPTH_TOLERANCE,
  @DRAFT_LIMIT,
  @DRAG_LIMIT,
  @TRIM_LIMIT,
  @TURN_LIMIT,
  @DRY_MATERIAL_MASS_DENSITY,
  @WATER_MASS_DENSITY,
  @DREDGE_AIDS,
  @COMMENT )
go
```

INSERT_PROJECT_SUMMARY_DATA

Purpose: *Insert a project summary record in the PROJECT_SUMMARY table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'INSERT_PROJECT_SUMMARY_DATA' )
  drop proc INSERT_PROJECT_SUMMARY_DATA
go
```

```
create proc INSERT_PROJECT_SUMMARY_DATA
( @CONTRACT_ID                      CONTRACT_IDENTIFIER,
  @PROJECT                         PROJECT_NAME,
  @DREDGE                          DREDGE_NAME,
  @OPENING_DATE                     datetime,
  @CLOSING_DATE                     datetime,
  @PUMPING_TIME                     MINUTES_TYPE,
  @TURNING_TIME                     MINUTES_TYPE,
  @SAILING_FULL_TIME                 MINUTES_TYPE,
  @DUMPING_TIME                     MINUTES_TYPE,
  @SAILING_EMPTY_TIME                MINUTES_TYPE,
  @NON_EFFECTIVE_TIME                MINUTES_TYPE,
  @LOST_TIME                       MINUTES_TYPE,
  @TO_BE_DEFINED_TIME                MINUTES_TYPE,
  @FUEL_AND_SUPPLIES_TIME             MINUTES_TYPE,
  @WHARF_OR_ANCHORAGE_TIME             MINUTES_TYPE,
  @OPPOSING_NATURAL_ELEMENT_TIME   MINUTES_TYPE,
  @TRAFFIC_AND_BRIDGES_TIME          MINUTES_TYPE,
  @MINOR_OPERATING_REPAIRS_TIME    MINUTES_TYPE,
  @TRANSFER_BETWEEN_WORKS_TIME     MINUTES_TYPE,
  @LAY_TIME                        MINUTES_TYPE,
  @FIRE_AND_BOAT_DRILLS_TIME         MINUTES_TYPE,
  @MISCELLANEOUS_TIME                MINUTES_TYPE,
  @MAJOR_REPAIRS_TIME                MINUTES_TYPE,
  @CESSATION_TIME                   MINUTES_TYPE,
  @COLLISIONS_TIME                  MINUTES_TYPE,
  @ESTIMATED_COST                    money,
  @JOB_TO_DATE_COST                   money,
  @COST_PER_MINUTE                   money,
  @TONS_RETAINED                    LONG_TONS_TYPE ) as
insert into PROJECT_SUMMARY
values
( @CONTRACT_ID,
  @PROJECT,
  @DREDGE,
  @OPENING_DATE,
  @CLOSING_DATE,
  @PUMPING_TIME,
  @TURNING_TIME,
  @SAILING_FULL_TIME,
  @DUMPING_TIME,
  @SAILING_EMPTY_TIME,
  @NON_EFFECTIVE_TIME,
  @LOST_TIME,
  @TO_BE_DEFINED_TIME,
  @FUEL_AND_SUPPLIES_TIME,
  @WHARF_OR_ANCHORAGE_TIME,
  @OPPOSING_NATURAL_ELEMENT_TIME,
  @TRAFFIC_AND_BRIDGES_TIME,
  @MINOR_OPERATING_REPAIRS_TIME,
  @TRANSFER_BETWEEN_WORKS_TIME,
  @LAY_TIME,
  @FIRE_AND_BOAT_DRILLS_TIME,
  @MISCELLANEOUS_TIME,
  @MAJOR_REPAIRS_TIME,
  @CESSATION_TIME,
  @COLLISIONS_TIME,
  @ESTIMATED_COST,
  @JOB_TO_DATE_COST,
  @COST_PER_MINUTE,
  @TONS_RETAINED )
go
```

INSERT_STATE

Purpose: Insert a state change record for a load in the STATE table.

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'INSERT_STATE' )
 drop proc INSERT_STATE
go
create proc INSERT_STATE
( @CONTRACT_ID           CONTRACT_IDENTIFIER,
 @PROJECT                PROJECT_NAME,
 @DREDGE                 DREDGE_NAME,
 @LOAD_NO                LOAD_NUMBER,
 @START_DATE_TIME        datetime,
 @VESSEL_X               COORDINATE_TYPE,
 @VESSEL_Y               COORDINATE_TYPE,
 @VESSEL_HEADING         DEGREES_TYPE,
 @VESSEL_SPEED           KNOTS_TYPE,
 @VESSEL_DRAFT_FORWARD   FEET_TYPE,
 @VESSEL_DRAFT_AFT       FEET_TYPE,
 @PORT_PUMP_STATUS       BOOLEAN_TYPE,
 @DRAGHEAD_DEPTH_PORT    FEET_TYPE,
 @DRAGHEAD_ELEVATION_PORT FEET_TYPE,
 @STBD_PUMP_STATUS       BOOLEAN_TYPE,
 @DRAGHEAD_DEPTH_STBD    FEET_TYPE,
 @DRAGHEAD_ELEVATION_STBD FEET_TYPE,
 @TIDE                   FEET_TYPE,
 @STATE                  DREDGE_STATE ) as
insert into STATE
values
( @CONTRACT_ID,
 @PROJECT,
 @DREDGE,
 @LOAD_NO,
 @START_DATE_TIME,
 @VESSEL_X,
 @VESSEL_Y,
 @VESSEL_HEADING,
 @VESSEL_SPEED,
 @VESSEL_DRAFT_FORWARD,
 @VESSEL_DRAFT_AFT,
 @PORT_PUMP_STATUS,
 @DRAGHEAD_DEPTH_PORT,
 @DRAGHEAD_ELEVATION_PORT,
 @STBD_PUMP_STATUS,
 @DRAGHEAD_DEPTH_STBD,
 @DRAGHEAD_ELEVATION_STBD,
 @TIDE,
 @STATE )
go
```

SELECT_DISPOSAL_AREA_DISTINCT

Purpose: Retrieve a unique disposal area having the specified AREA_NAME from the DISPOSAL_AREA table.

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'SELECT_DISPOSAL_AREA_DISTINCT' )
 drop proc SELECT_DISPOSAL_AREA_DISTINCT
go
```

```
create proc SELECT_DISPOSAL_AREA_DISTINCT ( @NAME AREA_NAME ) as
select BOUNDARY_01_X, BOUNDARY_01_Y, BOUNDARY_02_X, BOUNDARY_02_Y,
       BOUNDARY_03_X, BOUNDARY_03_Y, BOUNDARY_04_X, BOUNDARY_04_Y,
       BOUNDARY_05_X, BOUNDARY_05_Y, BOUNDARY_06_X, BOUNDARY_06_Y,
       BOUNDARY_07_X, BOUNDARY_07_Y, BOUNDARY_08_X, BOUNDARY_08_Y,
       BOUNDARY_09_X, BOUNDARY_09_Y, BOUNDARY_10_X, BOUNDARY_10_Y
from   DISPOSAL_AREA
where  NAME = @NAME
go
```

SELECT_DOWNTIME_DISTINCT

Purpose: *Retrieve a unique downtime event having the specified CONTRACT_ID, PROJECT, DREDGE and START_DATE_TIME from the DOWNTIME table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'SELECT_DOWNTIME_DISTINCT' )
  drop proc SELECT_DOWNTIME_DISTINCT
go
create proc SELECT_DOWNTIME_DISTINCT
( @CONTRACT_ID      CONTRACT_IDENTIFIER,
  @PROJECT          PROJECT_NAME,
  @DREDGE           DREDGE_NAME,
  @START_DATE_TIME datetime ) with recompile as
select END_DATE=convert( char(6), END_DATE_TIME, 12 ),
       END_TIME=convert( float, 60 * 60 * datepart(hh,END_DATE_TIME) +
                               60 * datepart(mi,END_DATE_TIME) +
                                    datepart(ss,END_DATE_TIME) +
                             .001 * datepart(ms,END_DATE_TIME) ),
       LOAD_NO, CAUSE, COMMENT
from   DOWNTIME
where  CONTRACT_ID          = @CONTRACT_ID
and    PROJECT              = @PROJECT
and    DREDGE               = @DREDGE
and    START_DATE_TIME      = @START_DATE_TIME
go
```

SELECT_LATEST_DREDGING_DATA

Purpose: *Retrieve the record having the latest date/time from the DREDGING_DATA table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'SELECT_LATEST_DREDGING_DATA' )
  drop proc SELECT_LATEST_DREDGING_DATA
go
create proc SELECT_LATEST_DREDGING_DATA with recompile as
select CONTRACT_ID,
       PROJECT,
       DREDGE,
       convert( char(6), DATE_TIME, 12 ),
       convert( float, 60 * 60 * datepart(hh,DATE_TIME) +
                           60 * datepart(mi,DATE_TIME) +
                                datepart(ss,DATE_TIME) +
                         .001 * datepart(ms,DATE_TIME) ),
       VESSEL_X,
       VESSEL_X_STATUS,
       VESSEL_Y,
       VESSEL_Y_STATUS,
      VESSEL_DRAFT_FORWARD,
      VESSEL_DRAFT_FORWARD_STATUS,
      VESSEL_DRAFT_AFT,
      VESSEL_DRAFT_AFT_STATUS,
      VESSEL_SPEED,
      VESSEL_SPEED_STATUS,
      VESSEL_HEADING,
      VESSEL_HEADING_STATUS,
      VESSEL_COURSE,
      VESSEL_COURSE_STATUS,
      DRAGHEAD_DEPTH_PORT,
      DRAGHEAD_DEPTH_PORT_STATUS,
      DRAGHEAD_DEPTH_STBD,
      DRAGHEAD_DEPTH_STBD_STATUS,
      HOPPER_LEVEL_FORWARD,
      HOPPER_LEVEL_FORWARD_STATUS,
      HOPPER_LEVEL_AFT,
      HOPPER_LEVEL_AFT_STATUS,
      HOPPER_VOLUME,
      HOPPER_VOLUME_STATUS,
      HOPPER_ULLAGE,
      HOPPER_ULLAGE_STATUS,
      WATER_DEPTH,
      WATER_DEPTH_STATUS,
      TIDE,
      TIDE_STATUS,
      HOPPER_DOOR_OPEN,
      PUMP_ON_PORT,
      PUMP_ON_STBD,
      PUMP_MATERIAL_PORT,
      PUMP_MATERIAL_STBD,
      PUMP_OUT_ON,
      LOAD_NO,
      STATE,
      TDM
from  DREDGING_DATA
where  DATE_TIME =
       ( select max(DATE_TIME)
         from  DREDGING_DATA )
go
```

SELECT_LOAD_DISTINCT

Purpose: *Retrieve a unique load record having the specified CONTRACT_ID, PROJECT, DREDGE and LOAD_NO from the LOAD table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
          name = 'SELECT_LOAD_DISTINCT' )
  drop proc SELECT_LOAD_DISTINCT
go
create proc SELECT_LOAD_DISTINCT
( @CONTRACT_ID     CONTRACT_IDENTIFIER,
  @PROJECT         PROJECT_NAME,
  @DREDGE          DREDGE_NAME,
  @LOAD_NO         LOAD_NUMBER ) with recompile as
select START_DATE=convert( char(6), DATE, 12 ),
       START_TIME=convert( float, 60 * 60 * datepart(hh,DATE) +
                                  60 * datepart(mi,DATE) +
                                       datepart(ss,DATE) +
                                  .001 * datepart(ms,DATE) ),
     PUMPING_TIME, TURNING_TIME, SAILING_FULL_TIME, DUMPING_TIME,
     SAILING_EMPTY_TIME, DOWN_TIME, TOTAL_TIME, TDM, COMMENT
from   LOAD_TABLE
where  CONTRACT_ID          = @CONTRACT_ID
and    PROJECT              = @PROJECT
and    DREDGE               = @DREDGE
and    LOAD_NO              = @LOAD_NO
go
```

SELECT_MIN_MAX_DATE_TIME

Purpose: *Retrieve the minimum and maximum date/times that exist in records in the DREDGING_DATA table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
          name = 'SELECT_MIN_MAX_DATE_TIME' )
  drop proc SELECT_MIN_MAX_DATE_TIME
go
create proc SELECT_MIN_MAX_DATE_TIME
( @CONTRACT_ID     CONTRACT_IDENTIFIER,
  @PROJECT         PROJECT_NAME,
  @DREDGE          DREDGE_NAME ) with recompile as
select MIN_DATE=convert(char(6),min(DATE_TIME),12),
     MIN_TIME=convert(float,60*60*datepart(hh,min(DATE_TIME)) +
                           60*datepart(mm,min(DATE_TIME)) +
                             datepart(ss,min(DATE_TIME)) +
                         0.001*datepart(ms,min(DATE_TIME))),
     MAX_DATE=convert(char(6),max(DATE_TIME),12),
     MAX_TIME=convert(float,60*60*datepart(hh,max(DATE_TIME)) +
                           60*datepart(mm,max(DATE_TIME)) +
                             datepart(ss,max(DATE_TIME)) +
                         0.001*datepart(ms,max(DATE_TIME)))
from   DREDGING_DATA
where  CONTRACT_ID          = @CONTRACT_ID
and    PROJECT              = @PROJECT
and    DREDGE               = @DREDGE
go
```

SELECT_PROCESSED_DREDGING_DATA

Purpose: *Retrieve the record having the latest date/time that has been processed (been assigned a state) by the RT_Kernel program from the DREDGING_DATA table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
          name = 'SELECT_PROCESSED_DREDGING_DATA' )
  drop proc SELECT_PROCESSED_DREDGING_DATA
go
create proc SELECT_PROCESSED_DREDGING_DATA with recompile as
select CONTRACT_ID,
     PROJECT,
     DREDGE,
     convert( char(6), DATE_TIME, 12 ),
     convert( float, 60 * 60 * datepart(hh,DATE_TIME) +
                     60 * datepart(mi,DATE_TIME) +
                          datepart(ss,DATE_TIME) +
                    .001 * datepart(ms,DATE_TIME) ),
     VESSEL_X,
     VESSEL_X_STATUS,
     VESSEL_Y,
     VESSEL_Y_STATUS,
       VESSEL_DRAFT_FORWARD,
       VESSEL_DRAFT_FORWARD_STATUS,
       VESSEL_DRAFT_AFT,
       VESSEL_DRAFT_AFT_STATUS,
       VESSEL_SPEED,
       VESSEL_SPEED_STATUS,
       VESSEL_HEADING,
       VESSEL_HEADING_STATUS,
       VESSEL_COURSE,
       VESSEL_COURSE_STATUS,
       DRAGHEAD_DEPTH_PORT,
       DRAGHEAD_DEPTH_PORT_STATUS,
       DRAGHEAD_DEPTH_STBD,
       DRAGHEAD_DEPTH_STBD_STATUS,
       HOPPER_LEVEL_FORWARD,
       HOPPER_LEVEL_FORWARD_STATUS,
       HOPPER_LEVEL_AFT,
       HOPPER_LEVEL_AFT_STATUS,
       HOPPER_VOLUME,
       HOPPER_VOLUME_STATUS,
       HOPPER_ULLAGE,
       HOPPER_ULLAGE_STATUS,
       WATER_DEPTH,
       WATER_DEPTH_STATUS,
       TIDE,
       TIDE_STATUS,
       HOPPER_DOOR_OPEN,
       PUMP_ON_PORT,
       PUMP_ON_STBD,
       PUMP_MATERIAL_PORT,
       PUMP_MATERIAL_STBD,
       PUMP_OUT_ON,
       LOAD_NO,
       STATE,
       TDM
from   DREDGING_DATA
where  DATE_TIME =
       ( select max(DATE_TIME)
         from   DREDGING_DATA
         where  STATE is not null )
go
```

SELECT_PROJECT_DISTINCT

Purpose: *Retrieve a unique project record having the specified CONTRACT_ID and PROJECT_NAME from the PROJECT table. Use a Unix join function with the DISTRICT Table to obtain the District name.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
              name = 'SELECT_PROJECT_DISTINCT' )
  drop proc SELECT_PROJECT_DISTINCT
go
create proc SELECT_PROJECT_DISTINCT
( @CONTRACT_ID CONTRACT_IDENTIFIER,
  @NAME        PROJECT_NAME ) as
select DISTRICT.NAME, THE_TYPE, CONTRACTOR,
       convert ( char(6) , START_DATE  , 12 ),
       convert ( char(6) , FINISH_DATE , 12 )
from   PROJECT, DISTRICT
where  CONTRACT_ID          = @CONTRACT_ID
and    PROJECT.NAME         = @NAME
and    DISTRICT.ABBREVIATION  = PROJECT.DISTRICT
go
```

SELECT_PROJ_SUMMARY_DISTINCT

Purpose: *Retrieve a unique project summary record having the specified CONTRACT_ID, PROJECT_NAME and DREDGE_NAME from the PROJECT_SUMMARY table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'SELECT_PROJ_SUMMARY_DISTINCT' )
  drop proc SELECT_PROJ_SUMMARY_DISTINCT
go
create proc SELECT_PROJ_SUMMARY_DISTINCT
( @CONTRACT_ID     CONTRACT_IDENTIFIER,
  @PROJECT         PROJECT_NAME,
  @DREDGE          DREDGE_NAME ) as
select OPENING_DATE=convert( char(6), OPENING_DATE, 12 ),
      CLOSING_DATE=convert( char(6), CLOSING_DATE, 12 ),
      PUMPING_TIME, TURNING_TIME, SAILING_FULL_TIME, DUMPING_TIME,
      SAILING_EMPTY_TIME, NON_EFFECTIVE_TIME, LOST_TIME,
      TO_BE_DEFINED_TIME, FUEL_AND_SUPPLIES_TIME, WHARF_OR_ANCHORAGE_TIME,
      OPPOSING_NATURAL_ELEMENTS_TIME, TRAFFIC_AND_BRIDGES_TIME,
      MINOR_OPERATING_REPAIRS_TIME, TRANSFER_BETWEEN_WORKS_TIME,
      LAY_TIME, FIRE_AND_BOAT_DRILLS_TIME, MISCELLANEOUS_TIME,
      MAJOR_REPAIRS_TIME, CESSATION_TIME, COLLISIONS_TIME,
      ESTIMATED_COST, JOB_TO_DATE_COST, COST_PER_MINUTE,
      TONS_RETAINED
from   PROJECT_SUMMARY
where  CONTRACT_ID          = @CONTRACT_ID
and    PROJECT              = @PROJECT
and    DREDGE               = @DREDGE
go
```

SELECT_STATION_DISTINCT

Purpose: *Retrieve a unique station (dredging area) record having the specified AREA_NAME from the STATION table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'SELECT_STATION_DISTINCT' )
  drop proc SELECT_STATION_DISTINCT
go
create proc SELECT_STATION_DISTINCT ( @NAME AREA_NAME ) as
select BOUNDARY_01_X, BOUNDARY_01_Y, BOUNDARY_02_X, BOUNDARY_02_Y,
      BOUNDARY_03_X, BOUNDARY_03_Y, BOUNDARY_04_X, BOUNDARY_04_Y,
      BOUNDARY_05_X, BOUNDARY_05_Y, BOUNDARY_06_X, BOUNDARY_06_Y,
      BOUNDARY_07_X, BOUNDARY_07_Y, BOUNDARY_08_X, BOUNDARY_08_Y,
      BOUNDARY_09_X, BOUNDARY_09_Y, BOUNDARY_10_X, BOUNDARY_10_Y
from   STATION
where  NAME = @NAME
go
```

SELECT_UNPROC_DREDGING_DATA

Purpose: *Retrieve the record having the earliest date/time that has not been processed (been assigned a state) by the RT_Kernel program from the DREDGING_DATA table.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
          name = 'SELECT_UNPROC_DREDGING_DATA' )
  drop proc SELECT_UNPROC_DREDGING_DATA
go
create proc SELECT_UNPROC_DREDGING_DATA with recompile as
select CONTRACT_ID,
       PROJECT,
       DREDGE,
       convert( char(6), DATE_TIME, 12 ),
       convert( float, 60 * 60 * datepart(hh,DATE_TIME) +
                       60 * datepart(mi,DATE_TIME) +
                          datepart(ss,DATE_TIME) +
                     .001 * datepart(ms,DATE_TIME) ),
       VESSEL_X,
       VESSEL_X_STATUS,
       VESSEL_Y,
       VESSEL_Y_STATUS,
       VESSEL_DRAFT_FORWARD,
       VESSEL_DRAFT_FORWARD_STATUS,
       VESSEL_DRAFT_AFT,
       VESSEL_DRAFT_AFT_STATUS,
       VESSEL_SPEED,
       VESSEL_SPEED_STATUS,
       VESSEL_HEADING,
       VESSEL_HEADING_STATUS,
       VESSEL_COURSE,
       VESSEL_COURSE_STATUS,
       DRAGHEAD_DEPTH_PORT,
       DRAGHEAD_DEPTH_PORT_STATUS,
       DRAGHEAD_DEPTH_STBD,
       DRAGHEAD_DEPTH_STBD_STATUS,
       HOPPER_LEVEL_FORWARD,
       HOPPER_LEVEL_FORWARD_STATUS,
       HOPPER_LEVEL_AFT,
       HOPPER_LEVEL_AFT_STATUS,
       HOPPER_VOLUME,
       HOPPER_VOLUME_STATUS,
       HOPPER_ULLAGE,
       HOPPER_ULLAGE_STATUS,
       WATER_DEPTH,
       WATER_DEPTH_STATUS,
       TIDE,
       TIDE_STATUS,
       HOPPER_DOOR_OPEN,
       PUMP_ON_PORT,
       PUMP_ON_STBD,
       PUMP_MATERIAL_PORT,
       PUMP_MATERIAL_STBD,
       PUMP_OUT_ON,
       LOAD_NO,
       STATE,
       TDM
from   DREDGING_DATA
where  DATE_TIME =
       ( select min(DATE_TIME)
         from   DREDGING_DATA
         where  STATE is null )
go
```

UPDATE_DOWNTIME

Purpose: *Update a downtime event in the DOWNTIME table with a cause and comment.*

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'UPDATE_DOWNTIME' )
 drop proc UPDATE_DOWNTIME
go
create proc UPDATE_DOWNTIME
 ( @CONTRACT_ID     CONTRACT_IDENTIFIER,
   @PROJECT         PROJECT_NAME,
   @DREDGE          DREDGE_NAME,
   @START_DATE_TIME datetime,
   @CAUSE           DOWNTIME_CAUSE,
   @COMMENT         FREE_FORM_TEXT ) with recompile as
update DOWNTIME
set    CAUSE   = @CAUSE,
       COMMENT = @COMMENT
where  CONTRACT_ID     = @CONTRACT_ID
and    PROJECT         = @PROJECT
and    DREDGE          = @DREDGE
and    START_DATE_TIME = @START_DATE_TIME
go
```

UPDATE_DREDGING_DATA

Purpose: Update a dredging data record in the DREDGING_DATA table with a load number, state and TDM.

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'UPDATE_DREDGING_DATA' )
 drop proc UPDATE_DREDGING_DATA
go
create proc UPDATE_DREDGING_DATA
 ( @CONTRACT_ID CONTRACT_IDENTIFIER,
   @PROJECT     PROJECT_NAME,
   @DREDGE      DREDGE_NAME,
   @DATE_TIME   datetime,
   @LOAD_NO     LOAD_NUMBER,
   @STATE       DREDGE_STATE,
   @TDM         LONG_TONS_TYPE ) with recompile as
update DREDGING_DATA
set    LOAD_NO = @LOAD_NO,
       STATE   = @STATE,
       TDM     = @TDM
where  CONTRACT_ID = @CONTRACT_ID
and    PROJECT     = @PROJECT
and    DREDGE      = @DREDGE
and    DATE_TIME   = @DATE_TIME
go
```

UPDATE_PROJECT_SUMMARY

Purpose: Update a project summary record in the PROJECT_SUMMARY table.

Code:

```
if exists ( select * from sysobjects where uid = user_id() and
            name = 'UPDATE_PROJECT_SUMMARY' )
 drop proc UPDATE_PROJECT_SUMMARY
go
create proc UPDATE_PROJECT_SUMMARY
 ( @CONTRACT_ID                CONTRACT_IDENTIFIER,
   @PROJECT                   PROJECT_NAME,
   @DREDGE                    DREDGE_NAME,
    @CLOSING_DATE                  datetime,
    @PUMPING_TIME                  MINUTES_TYPE,
    @TURNING_TIME                  MINUTES_TYPE,
    @SAILING_FULL_TIME               MINUTES_TYPE,
    @DUMPING_TIME                  MINUTES_TYPE,
    @SAILING_EMPTY_TIME              MINUTES_TYPE,
    @NON_EFFECTIVE_TIME              MINUTES_TYPE,
    @LOST_TIME                   MINUTES_TYPE,
    @TO_BE_DEFINED_TIME              MINUTES_TYPE,
    @FUEL_AND_SUPPLIES_TIME            MINUTES_TYPE,
    @WHARF_OR_ANCHORAGE_TIME           MINUTES_TYPE,
    @OPPOSING_NATURAL_ELEMENT_TIME   MINUTES_TYPE,
    @TRAFFIC_AND_BRIDGES_TIME         MINUTES_TYPE,
    @MINOR_OPERATING_REPAIRS_TIME    MINUTES_TYPE,
    @TRANSFER_BETWEEN_WORKS_TIME     MINUTES_TYPE,
    @LAY_TIME                    MINUTES_TYPE,
    @FIRE_AND_BOAT_DRILLS_TIME        MINUTES_TYPE,
    @MISCELLANEOUS_TIME              MINUTES_TYPE,
    @MAJOR_REPAIRS_TIME              MINUTES_TYPE,
    @CESSATION_TIME                 MINUTES_TYPE,
    @COLLISIONS_TIME                MINUTES_TYPE,
    @ESTIMATED_COST                  money,
    @JOB_TO_DATE_COST                 money,
    @COST_PER_MINUTE                  money,
    @TONS_RETAINED                 LONG_TONS_TYPE ) as
update PROJECT_SUMMARY
set    CLOSING_DATE                  = @CLOSING_DATE,
       PUMPING_TIME                  = @PUMPING_TIME,
       TURNING_TIME                  = @TURNING_TIME,
       SAILING_FULL_TIME               = @SAILING_FULL_TIME,
       DUMPING_TIME                  = @DUMPING_TIME,
       SAILING_EMPTY_TIME              = @SAILING_EMPTY_TIME,
       NON_EFFECTIVE_TIME              = @NON_EFFECTIVE_TIME,
       LOST_TIME                   = @LOST_TIME,
       TO_BE_DEFINED_TIME              = @TO_BE_DEFINED_TIME,
       FUEL_AND_SUPPLIES_TIME           = @FUEL_AND_SUPPLIES_TIME,
       WHARF_OR_ANCHORAGE_TIME          = @WHARF_OR_ANCHORAGE_TIME,
       OPPOSING_NATURAL_ELEMENTS_TIME = @OPPOSING_NATURAL_ELEMENT_TIME,
       TRAFFIC_AND_BRIDGES_TIME         = @TRAFFIC_AND_BRIDGES_TIME,
       MINOR_OPERATING_REPAIRS_TIME   = @MINOR_OPERATING_REPAIRS_TIME,
       TRANSFER_BETWEEN_WORKS_TIME    = @TRANSFER_BETWEEN_WORKS_TIME,
       LAY_TIME                    = @LAY_TIME,
       FIRE_AND_BOAT_DRILLS_TIME        = @FIRE_AND_BOAT_DRILLS_TIME,
       MISCELLANEOUS_TIME              = @MISCELLANEOUS_TIME,
       MAJOR_REPAIRS_TIME              = @MAJOR_REPAIRS_TIME,
       CESSATION_TIME                 = @CESSATION_TIME,
       COLLISIONS_TIME                = @COLLISIONS_TIME,
       ESTIMATED_COST                  = @ESTIMATED_COST,
       JOB_TO_DATE_COST                 = @JOB_TO_DATE_COST,
       COST_PER_MINUTE                  = @COST_PER_MINUTE,
       TONS_RETAINED                 = @TONS_RETAINED
where  CONTRACT_ID     = @CONTRACT_ID
and    PROJECT         = @PROJECT
and    DREDGE          = @DREDGE
go
```

```
    @JOB_TO_DATE_COST             money,
    @COST_PER_MINUTE              money,
    @TONS_RETAINED                LONG_TONS_TYPE ) as
update PROJECT_SUMMARY
set   CLOSING_DATE                    = @CLOSING_DATE,
      PUMPING_TIME                    = @PUMPING_TIME,
      TURNING_TIME                    = @TURNING_TIME,
      SAILING_FULL_TIME               = @SAILING_FULL_TIME,
      DUMPING_TIME                    = @DUMPING_TIME,
      SAILING_EMPTY_TIME              = @SAILING_EMPTY_TIME,
      NON_EFFECTIVE_TIME              = @NON_EFFECTIVE_TIME,
      LOST_TIME                       = @LOST_TIME,
      TO_BE_DEFINED_TIME              = @TO_BE_DEFINED_TIME,
      FUEL_AND_SUPPLIES_TIME          = @FUEL_AND_SUPPLIES_TIME,
      WHARF_OR_ANCHORAGE_TIME         = @WHARF_OR_ANCHORAGE_TIME,
      OPPOSING_NATURAL_ELEMENTS_TIME = @OPPOSING_NATURAL_ELEMENT_TIME,
      TRAFFIC_AND_BRIDGES_TIME        = @TRAFFIC_AND_BRIDGES_TIME,
      MINOR_OPERATING_REPAIRS_TIME    = @MINOR_OPERATING_REPAIRS_TIME,
      TRANSFER_BETWEEN_WORKS_TIME     = @TRANSFER_BETWEEN_WORKS_TIME,
      LAY_TIME                        = @LAY_TIME,
      FIRE_AND_BOAT_DRILLS_TIME       = @FIRE_AND_BOAT_DRILLS_TIME,
      MISCELLANEOUS_TIME              = @MISCELLANEOUS_TIME,
      MAJOR_REPAIRS_TIME              = @MAJOR_REPAIRS_TIME,
      CESSATION_TIME                  = @CESSATION_TIME,
      COLLISIONS_TIME                 = @COLLISIONS_TIME,
      ESTIMATED_COST                  = @ESTIMATED_COST,
      JOB_TO_DATE_COST                = @JOB_TO_DATE_COST,
      COST_PER_MINUTE                 = @COST_PER_MINUTE,
      TONS_RETAINED                   = @TONS_RETAINED
where  CONTRACT_ID    = @CONTRACT_ID
and    PROJECT        = @PROJECT
and    DREDGE         = @DREDGE
go
```

Appendix B
Backup and Archive Script

ARCHIVE FUNCTION SCRIPTING

```
#!csh
#
# dosis_archive - write archive files to $DOSIS_ARCHIVE, remove archived data
#                 from the database
#
unset noclobber
#
# make sure all required environment variables are set correctly
#
if ( $?DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR not set; please set and rerun"
  set DOSIS_COMMAND_DIR = dummy
  exit(1)
else if ( -d $DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR does not designate a directory"
  echo "      Please set and rerun"
  exit(1)
endif
if ( -e $DOSIS_COMMAND_DIR/check_vars == 0 ) then
  echo "$DOSIS_COMMAND_DIR/check_vars script does not exist"
  echo "      Please fix and rerun"
  exit(1)
endif
source $DOSIS_COMMAND_DIR/check_vars
if ( $status != 0 ) exit(1)
#
# make sure an archive hasn't already been taken today
#
if ( -e $DOSIS_ARCHIVE/$date$ArchiveTables[2].ad == 1 ) then
  echo "An archive has already been taken today.  Please wait for tomorrow"
  exit(1)
endif
#
# dump the transaction log before starting the archive
#
if ( -e $DOSIS_COMMAND_DIR/dosis_dumptran == 0 || \
     -e $DOSIS_COMMAND_DIR/dosis_dumpdb   == 0 ) then
echo "Can't find $DOSIS_COMMAND_DIR/dosis_dumptran and/or dosis_dumpdb scripts"
  echo "      Please fix and rerun"
  exit(1)
endif
```

```
source $DOSIS_COMMAND_DIR/dosis_dumptran
if ( $status != 0 ) exit(1)
#
# set options to allow us to select into, etc.
#
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
sp_dboption DOSIS, selec, true
go
sp_dboption DOSIS, trunc, true
go
use DOSIS
go
checkpoint
go
EXIT
diff DOSIStemp - >/dev/null <<EXIT
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
Run the CHECKPOINT command in the database that was changed.
(return status = 0)

EXIT
if ( $status != 0 ) then
  echo "Unable to set database options: please correct below errors and rerun"
  cat DOSIStemp
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
#
# bulk copy the required tables
#
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
use DOSIS
go
select "marker", max(LOAD_NO)
from   LOAD_TABLE
go
EXIT
set return = `grep marker DOSIStemp`
if ( $#return == 0 ) then
  echo "Can't read max load number; please correct below errors and rerun"
  cat DOSIStemp
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
set MaxLoadNo = $return[2]
if ( $MaxLoadNo == "NULL" ) then
  echo "No loads have been completed; please rerun archive request later"
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
set WorkArchiveTables = ( $ArchiveTables )
while ( $#WorkArchiveTables != 0 )
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
use DOSIS
go
if exists ( select * from sysobjects where name = "ArchiveTemp" )
  drop table ArchiveTemp
go
select *
into   ArchiveTemp
from   $WorkArchiveTables[1]
```

```
where  LOAD_NO <= $MaxLoadNo
go
declare @OriginalCount int, @CopiedCount int
select @OriginalCount = count(*)
from   $WorkArchiveTables[1]
where  LOAD_NO <= $MaxLoadNo
select @CopiedCount = count(*)
from   ArchiveTemp
if ( @OriginalCount = @CopiedCount )
  select "SUCCESS", @CopiedCount
go
EXIT
set return = `grep SUCCESS DOSIStemp`
if ( $#return == 0 ) then
  echo "Could not copy table for bulk copy"
  echo "     Please correct below errors and rerun"
  cat DOSIStemp
  rm $DOSIS_ARCHIVE/$date*.ad
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
set RowCount = $return[2]
bcp DOSIS.dbo.ArchiveTemp out $DOSIS_ARCHIVE/$date$WorkArchiveTables[2].ad \
  -n -Usa -P$SA_PASSWORD >DOSIStemp
set return = `grep "rows copied\." DOSIStemp`
if ( $#return == 0 ) then
  echo "Bulk copy out failed - please correct below errors and rerun"
  cat DOSIStemp
  rm $DOSIS_ARCHIVE/$date*.ad
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
if ( $return[1] != $RowCount ) then
  echo "Wrong row count on bulk copy out - please check below errors and rerun"
  cat DOSIStemp
  rm $DOSIS_ARCHIVE/$date*.ad
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
shift WorkArchiveTables
shift WorkArchiveTables
shift WorkArchiveTables
end
#
# delete the archived data
#
set WorkArchiveTables = ( $ArchiveTables )
while ( $#WorkArchiveTables != 0 )
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
use DOSIS
go
delete $WorkArchiveTables[1]
where  LOAD_NO <= $MaxLoadNo
go
EXIT
set return = `grep affected DOSIStemp`
set return1 = `grep Msg DOSIStemp`
if ( $#return == 0 || $#return1 != 0 ) then
  echo "*****        DELETE FAILED FOR ARCHIVED DATA         *****"
  echo "*****   YOUR DOSIS DATABASE MAY NOW BE INCONSISTENT   *****"
  echo "***** WE STRONGLY RECOMMEND RUNNING dosis_restore NOW *****"
  echo "*****              see below errors                  *****"
  cat DOSIStemp
```

```
endif
shift WorkArchiveTables
shift WorkArchiveTables
shift WorkArchiveTables
end
#
# Done - report success, dump the database
#
echo "Archive files for $date have been created"
csh $DOSIS_COMMAND_DIR/dosis_dumpdb
exit(0)
```

CHECKING FUNCTION FOR ARCHIVING SCRIPTING

```
#!csh
#
# check_vars - make sure all environment variables are set for DOSIS scripts
#
# tables to archive
#
set ArchiveTables = ( \
  DREDGING_DATA      DDA DATE_TIME \
  STATE              STA START_DATE_TIME \
  LOAD_TABLE         LDT DATE \
  DOWNTIME           DWN START_DATE_TIME \
  LOAD_DISPOSAL_AREA LDA DISPOSAL_START_DATE_TIME \
  LOAD_STATION       LST STATION_START_DATE_TIME )
#
# make sure all required environment variables are set correctly
#
set DiskDumpDevice = 2
unset noclobber
set error = 0
if ( $?SA_PASSWORD == 0 ) then
  echo "Environment variable SA_PASSWORD not set; please set and rerun"
  set error = 1
else
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
select 1234
go
EXIT
set return=`grep 1234 DOSIStemp`
if ( $return != 1234 ) then
  echo "Environment variable SA_PASSWORD is not correct sa password"
  echo "     Please set and rerun"
  set error = 1
endif
endif
if ( $?DOSIS_ARCHIVE == 0 ) then
  echo "Environment variable DOSIS_ARCHIVE not set; please set and rerun"
  set DOSIS_ARCHIVE = dummy
  set error = 1
else if ( -d $DOSIS_ARCHIVE == 0 ) then
  echo "Environment variable DOSIS_ARCHIVE does not designate a directory"
  echo "     Please set and rerun"
  set error = 1
endif
if ( $?DOSIS_BACKUP == 0 ) then
  echo "Environment variable DOSIS_BACKUP not set; please set and rerun"
  set DOSIS_BACKUP = dummy
  set error = 1
else if ( -d $DOSIS_BACKUP == 0 ) then
```

```
  echo "Environment variable DOSIS_BACKUP does not designate a directory"
  echo "      Please set and rerun"
  set error = 1
endif
if ( $?DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR not set; please set and rerun"
  set DOSIS_COMMAND_DIR = dummy
  set error = 1
else if ( -d $DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR does not designate a directory"
  echo "      Please set and rerun"
  set error = 1
endif
if ( $?DOSIS_DUMP_DEVICE == 0 ) then
  echo "Environment variable DOSIS_DUMP_DEVICE not set; please set and rerun"
  set error = 1
else
isql -Usa -P$SA_PASSWORD -w256 <<EXIT >DOSIStemp
select "marker", cntrltype, phyname
from   sysdevices
where  name = "$DOSIS_DUMP_DEVICE"
go
EXIT
set return = `grep marker DOSIStemp`
if ( $#return == 0 ) then
  echo "Environment variable DOSIS_DUMP_DEVICE not set to a valid dump device"
  echo "      Please set and rerun"
  set return = ( 2 2 2 )
  set error = 1
endif
if ( $return[2] != $DiskDumpDevice ) then
  echo "Environment variable DOSIS_DUMP_DEVICE not set to a valid dump device"
  echo "      Please set and rerun"
  set error = 1
endif
set DOSIS_DUMP_FILE = $return[3]
endif
if ( $error == 1 ) exit(1)
#
# establish date for file names and archive time
#
set date=`date +'%y%m%d'`
if ( $date < '500000' ) then
  set date = 20${date}
else
  set date = 19${date}
endif
exit(0)
```

DELETE ARCHIVED FILES FUNCTION SCRIPTING

```
#!csh
#
# dosis_clean [ yyyymmdd ] - delete archive files from yyyymmdd (if specified),
#                            else oldest archive set
#
unset noclobber
#
# make sure all required environment variables are set correctly
#
if ( $?DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR not set; please set and rerun"
  set DOSIS_COMMAND_DIR = dummy
  exit(1)
else if ( -d $DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR does not designate a directory"
  echo "     Please set and rerun"
  exit(1)
endif
if ( -e $DOSIS_COMMAND_DIR/check_vars == 0 ) then
  echo "$DOSIS_COMMAND_DIR/check_vars script does not exist"
  echo "     Please fix and rerun"
  exit(1)
endif
source $DOSIS_COMMAND_DIR/check_vars
if ( $status != 0 ) exit(1)
#
# get date either from (a) our argument or (b) oldest archive file
#
if ( $#argv == 0 ) then
  set Date = ( $DOSIS_COMMAND_DIR/check*_vars $DOSIS_ARCHIVE/*.ad )
  shift Date
  if ( $#Date == 0 ) then
    echo "No archive files are present.  No need to clean them out"
    exit(1)
  endif
  awk '{ print substr($1,1,8) }' <<EXIT >DOSIStemp
$Date[1]:t
EXIT
  set Date = `cat DOSIStemp`
else
  set Date = $argv[1]
endif
#
# make sure at least one archive files exists
#
set WorkArchiveTables = ( $ArchiveTables )
unset exists
while ( $#WorkArchiveTables != 0 )
  set filename = $DOSIS_ARCHIVE/$Date$WorkArchiveTables[2].ad
  if ( -e $filename ) then
    set exists = 1
    rm $filename
  endif
  shift WorkArchiveTables
  shift WorkArchiveTables
  shift WorkArchiveTables
end
if ( $?exists == 0 ) then
  echo "No archive files found for that date.  Please rerun with correct date"
  exit(1)
endif
echo "Archive files for $Date deleted"
exit(0)
```

DUMP DOSIS DATABASE FUNCTION SCRIPTING

```
#!csh
#
# dosis_dumpdb - dump DOSIS database, also delete all but latest database and
#               transaction log dumps
#
unset noclobber
#
```

```
# make sure all required environment variables are set correctly
#
if ( $?DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR not set; please set and rerun"
  set DOSIS_COMMAND_DIR = dummy
  exit(1)
else if ( -d $DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR does not designate a directory"
  echo "     Please set and rerun"
  exit(1)
endif
if ( -e $DOSIS_COMMAND_DIR/check_vars == 0 ) then
  echo "$DOSIS_COMMAND_DIR/check_vars script does not exist"
  echo "     Please fix and rerun"
  exit(1)
endif
source $DOSIS_COMMAND_DIR/check_vars
if ( $status != 0 ) exit(1)
#
# delete old backups
#
set DatabaseDumps = ( $DOSIS_COMMAND_DIR/check*_vars $DOSIS_BACKUP/*.dbd )
shift DatabaseDumps
while ( $#DatabaseDumps > 1 )
  set LogDumps = ( $DOSIS_COMMAND_DIR/check*_vars $DOSIS_BACKUP/*.tld )
  shift LogDumps
  while ( $#LogDumps > 0 )
    sort <<EXIT >DOSIStemp
$LogDumps[1]:r
$DatabaseDumps[2]:r
EXIT
    set return = 'cat DOSIStemp'
    if ( $return[1] == $LogDumps[1]:r ) then
      rm $LogDumps[1]
    else
      break
    endif
    shift LogDumps
  end
  rm $DatabaseDumps[1]
  shift DatabaseDumps
end
#
# find the suffix for the database dump
#
if ( -e $DOSIS_BACKUP/${date}z.dbd ) then
  echo "Too many dumps taken today.  Please check $DOSIS_BACKUP"
  exit(1)
endif
set prev_suffix = z
foreach suffix ( y x w v u t s r q p o n m l k j i h g f e d c b a )
  if ( -e $DOSIS_BACKUP/$date$suffix.dbd ) then
    set suffix = $prev_suffix
    break
  endif
  if ( -e $DOSIS_BACKUP/$date$suffix.tld ) break
  set prev_suffix = $suffix
end
#
# set options and truncate the transaction log in preparation for dumping
#
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
sp_dboption DOSIS, selec, false
```

```
go
sp_dboption DOSIS, trunc, false
go
use DOSIS
go
checkpoint
go
dump transaction DOSIS with truncate_only
go
EXIT
diff DOSIStemp - >/dev/null <<EXIT
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
Run the CHECKPOINT command in the database that was changed.
(return status = 0)

EXIT
if ( $status != 0 ) then
  echo "Unable to set database options and truncate transaction log"
  echo "      Please correct below errors and rerun"
  cat DOSIStemp
  exit(1)
endif
#
# dump the database
#
if ( -e $DOSIS_DUMP_FILE ) rm $DOSIS_DUMP_FILE
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
dump database DOSIS to $DOSIS_DUMP_DEVICE
go
EXIT
set return = `cat DOSIStemp`
if ( $#return != 0 || -e $DOSIS_DUMP_FILE == 0 ) then
  echo "Database dump failed -- please correct below errors and rerun"
  cat DOSIStemp
  exit(1)
endif
set filename = $DOSIS_BACKUP/$date$suffix.dbd
mv $DOSIS_DUMP_FILE $filename
if ( -e $filename != 1 ) then
  echo "Failed to copy database dump -- please correct and rerun"
  exit(1)
endif
#
# success message
#
echo "DOSIS database dumped to $filename"
exit(0)
```

DUMP DOSIS DATABASE TRANSACTION LOG FUNCTION SCRIPTING

```
#!csh
#
# dosis_dumptran - dump DOSIS database transaction log
#
unset noclobber
#
# make sure all required environment variables are set correctly
#
if ( $?DOSIS_COMMAND_DIR == 0 ) then
```

```
  echo "Environment variable DOSIS_COMMAND_DIR not set; please set and rerun"
  set DOSIS_COMMAND_DIR = dummy
  exit(1)
else if ( -d $DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR does not designate a directory"
  echo "      Please set and rerun"
  exit(1)
endif
if ( -e $DOSIS_COMMAND_DIR/check_vars == 0 ) then
  echo "$DOSIS_COMMAND_DIR/check_vars script does not exist"
  echo "      Please fix and rerun"
  exit(1)
endif
source $DOSIS_COMMAND_DIR/check_vars
if ( $status != 0 ) exit(1)
#
# find the suffix for the transaction log
#
if ( -e $DOSIS_BACKUP/${date}z.dbd || -e $DOSIS_BACKUP/${date}z.tld ) then
  echo "Too many dumps taken today.  Please check $DOSIS_BACKUP"
  exit(1)
endif
set prev_suffix = z
foreach suffix ( y x w v u t s r q p o n m l k j i h g f e d c b a )
  if ( -e $DOSIS_BACKUP/$date$suffix.dbd || \
       -e $DOSIS_BACKUP/$date$suffix.tld ) then
    set suffix = $prev_suffix
    break
  endif
  set prev_suffix = $suffix
end
#
# dump the transaction log
#
if ( -e $DOSIS_DUMP_FILE == 1 ) rm $DOSIS_DUMP_FILE
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
dump transaction DOSIS to $DOSIS_DUMP_DEVICE
go
EXIT
set return = `cat DOSIStemp`
if ( $#return != 0 || -e $DOSIS_DUMP_FILE == 0 ) then
  echo "Transaction log dump failed -- please correct below errors and rerun"
  cat DOSIStemp
  exit(1)
endif
set filename = $DOSIS_BACKUP/$date$suffix.tld
mv $DOSIS_DUMP_FILE $filename
if ( -e $filename != 1 ) then
  echo "Failed to copy transaction log dump -- please correct and rerun"
  exit(1)
endif
#
# report success
#
echo "DOSIS transaction log dumped to $filename"
exit(0)
```

LOAD ARCHIVED DATA FUNCTION SCRIPTING

```
#!csh
#
# dosis_load [ DEV ] - load archive from /dev/DEV (if specified),
#                    else from /dev/$DOSIS_TAPE
#
unset noclobber
#
# make sure all required environment variables are set correctly
#
if ( $?DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR not set; please set and rerun"
  set DOSIS_COMMAND_DIR = dummy
  exit(1)
else if ( -d $DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR does not designate a directory"
  echo "     Please set and rerun"
  exit(1)
endif
if ( -e $DOSIS_COMMAND_DIR/check_vars == 0 ) then
  echo "$DOSIS_COMMAND_DIR/check_vars script does not exist"
  echo "     Please fix and rerun"
  exit(1)
endif
source $DOSIS_COMMAND_DIR/check_vars
if ( $status != 0 ) exit(1)
#
# use argument for tape device if specified, else $DOSIS_TAPE
#
if ( $#argv == 0 ) then
  if ( $?DOSIS_TAPE == 0 ) then
    echo "You must specify a tape drive, or set DOSIS_TAPE environment var."
    echo "     Please correct and rerun"
    exit(1)
  endif
  set Tape = /dev/$DOSIS_TAPE
else
  set Tape = /dev/$argv[1]
endif
#
# make sure Tape device is valid
#
if ( -e $Tape == 0 ) then
  echo "Tape device $Tape does not exist.  Please check and rerun"
  exit(1)
endif
#
# load the archived files onto disk
#
( cd $DOSIS_ARCHIVE ; tar xvf $Tape >&DOSIStemp )
if ( $status != 0 ) then
  echo "Cannot read archive tape; please correct below errors and rerun"
  cat DOSIStemp
  exit(1)
endif
#
# get the file names; make sure we have the right number
#
set filenames = `awk '/^x/ { print substr($2,1,14) }' DOSIStemp`
if ( 3 * $#filenames != $#ArchiveTables ) then
  echo "Wrong number of files read from tape - please correct and rerun"
  foreach file ( $filenames )
    rm $DOSIS_ARCHIVE/$file
  end
  exit(1)
endif
```

```
#
# get the date from the file names
#
set Date = `awk '/^x/ { print substr($2,1,8) }' DOSIStemp`
while ( $#Date > 1 )
  if ( $Date[1] != $Date[2] ) then
    echo "Files read from tape have different dates.  Please correct and rerun"
    foreach file ( $filenames )
      rm $DOSIS_ARCHIVE/$file
    end
    exit(1)
  endif
  shift Date
end
#
# verify that all required files have been read in
#
set WorkArchiveTables = ( $ArchiveTables )
unset missing
while ( $#WorkArchiveTables != 0 )
  set filename = $Date$WorkArchiveTables[2].ad
  set return = `grep $filename DOSIStemp`
  if ( $#return == 0 ) then
    echo "File $filename missing from tape"
    set missing = 1
  endif
  shift WorkArchiveTables
  shift WorkArchiveTables
  shift WorkArchiveTables .
end
if ( $?missing ) then
  echo "Please correct above errors and rerun"
  foreach file ( $filenames )
    rm $DOSIS_ARCHIVE/$file
  end
  exit(1)
endif
#
# dump the transaction log before starting the load (since it
involves bcp)
#
if ( -e $DOSIS_COMMAND_DIR/dosis_dumptran == 0 || \
     -e $DOSIS_COMMAND_DIR/dosis_dumpdb   == 0 ) then
echo "Can't find $DOSIS_COMMAND_DIR/dosis_dumptran and/or dosis_dumpdb scripts"
  echo "     Please fix and rerun"
  foreach file ( $filenames )
    rm $DOSIS_ARCHIVE/$file
  end
  exit(1)
endif
source $DOSIS_COMMAND_DIR/dosis_dumptran
if ( $status != 0 ) then
  foreach file ( $filenames )
    rm $DOSIS_ARCHIVE/$file
  end
  exit(1)
endif
#
# set options to allow us to bulk copy in
#
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
sp_dboption DOSIS, selec, true
go
```

```
sp_dboption DOSIS, trunc, true
go
use DOSIS
go
checkpoint
go
EXIT
diff DOSIStemp - >/dev/null <<EXIT
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
Run the CHECKPOINT command in the database that was changed.
(return status = 0)

EXIT
if ( $status != 0 ) then
  echo "Unable to set database options: please correct below errors and rerun"
  cat DOSIStemp
  foreach file ( $filenames )
    rm $DOSIS_ARCHIVE/$file
  end
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
#
# create a dummy table to hold the LOAD_TABLE
#
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
use DOSIS
go
if exists ( select * from sysobjects where name = "ArchiveTemp" )
  drop table ArchiveTemp
go
select *
into   ArchiveTemp
from   LOAD_TABLE
where  CONTRACT_ID = 'This will match no IDs'
go
EXIT
diff DOSIStemp - >/dev/null <<EXIT
(0 rows affected)

EXIT
if ( $status != 0 ) then
  echo "Unable to create LOAD_TABLE temporary table for analysis"
  echo "     Please correct below errors and rerun"
  cat DOSIStemp
  foreach file ( $filenames )
    rm $DOSIS_ARCHIVE/$file
  end
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
#
# bulk copy in the load table
#
bcp DOSIS.dbo.ArchiveTemp in $DOSIS_ARCHIVE/${Date}LDT.ad -n -Usa \
  -P$SA_PASSWORD >DOSIStemp
set return = `grep Msg DOSIStemp`
set return1 = `grep "rows copied" DOSIStemp`
if ( $#return1 == 0 ) set return1 = ( 0 0 )
if ( $#return != 0 || $return1[1] == 0 ) then
```

```
  echo "Bulk copy in of LOAD_TABLE failed or zero rows read"
  echo "     Please correct below errors and rerun"
  cat DOSIStemp
  foreach file ( $filenames )
    rm $DOSIS_ARCHIVE/$file
  end
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
#
# make sure we are not duplicating data
#
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
use DOSIS
go
delete ArchiveTemp
where  LOAD_NO <
  ( select max(LOAD_NO)
    from   ArchiveTemp )
go
select "marker", count(*)
from   ArchiveTemp a, LOAD_TABLE l
where  a.CONTRACT_ID = l.CONTRACT_ID
and    a.PROJECT     = l.PROJECT
and    a.DREDGE      = l.DREDGE
and    a.LOAD_NO     = l.LOAD_NO
go
EXIT
set return = `grep marker DOSIStemp`
set return1 = `grep Msg DOSIStemp`
if ( $#return == 0 || $#return1 != 0 ) then
  echo "Check for duplicate data being loaded from archive failed"
  echo "     Please correct below errors and rerun"
  cat DOSIStemp
  foreach file ( $filenames )
    rm $DOSIS_ARCHIVE/$file
  end
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
if ( $return[2] != 0 ) then
  echo "Duplicate data being inserted by load - please correct and rerun"
  foreach file ( $filenames )
    rm $DOSIS_ARCHIVE/$file
  end
  csh $DOSIS_COMMAND_DIR/dosis_dumpdb
  exit(1)
endif
#
# now bulk copy the data
#
set WorkArchiveTables = ( $ArchiveTables )
while ( $#WorkArchiveTables != 0 )
  bcp DOSIS.dbo.$WorkArchiveTables[1] in \
    $DOSIS_ARCHIVE/$Date$WorkArchiveTables[2].ad -n -Usa -P$SA_PASSWORD \
    >DOSIStemp
  set return = `grep "Msg|fail" DOSIStemp`
  set return1 = `grep "rows copied" DOSIStemp`
  if ( $#return != 0 || $#return1 == 0 ) then
    echo "*****   BULK COPY IN FAILED FOR LOADING AN ARCHIVE    *****"
    echo "*****   YOUR DOSIS DATABASE MAY NOW BE INCONSISTENT   *****"
    echo "***** WE STRONGLY RECOMMEND RUNNING dosis_restore NOW *****"
    echo "*****             see below errors                  *****"
    cat DOSIStemp
    foreach file ( $filenames )
      rm $DOSIS_ARCHIVE/$file
    end
    exit(1)
  endif
  shift WorkArchiveTables
  shift WorkArchiveTables
  shift WorkArchiveTables
end
#
# Done - report success, dump the database
#
echo "Archive files for $Date have been loaded"
foreach file ( $filenames )
  rm $DOSIS_ARCHIVE/$file
end
csh $DOSIS_COMMAND_DIR/dosis_dumpdb
exit(0)
```

RESTORE DOSIS DATABASE FUNCTION SCRIPTING

```
#!csh
#
# dosis_restore - restore latest DOSIS database and transaction log dumps
#
unset noclobber
#
# make sure all required environment variables are set correctly
#
if ( $?DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR not set; please set and rerun"
  set DOSIS_COMMAND_DIR = dummy
  exit(1)
else if ( -d $DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR does not designate a directory"
  echo "      Please set and rerun"
  exit(1)
endif
if ( -e $DOSIS_COMMAND_DIR/check_vars == 0 ) then
  echo "$DOSIS_COMMAND_DIR/check_vars script does not exist"
  echo "      Please fix and rerun"
  exit(1)
endif
source $DOSIS_COMMAND_DIR/check_vars
if ( $status != 0 ) exit(1)
#
# find latest database dump
#
set DatabaseDumps = ( $DOSIS_COMMAND_DIR/check*_vars $DOSIS_BACKUP/*.dbd )
shift DatabaseDumps
if ( $#DatabaseDumps == 0 ) then
  echo "No DOSIS database dumps available; cannot restore!"
  exit(1)
endif
while ( $#DatabaseDumps > 1 )
  shift DatabaseDumps
end
#
# load database dump
```

```
#
cp $DatabaseDumps $DOSIS_DUMP_FILE
if ( $status != 0 ) then
  echo "Copy of database dump $DatabaseDumps failed"
  echo "     Please correct and rerun"
  exit(1)
endif
isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
load database DOSIS from $DOSIS_DUMP_DEVICE
go
EXIT
set return = `cat DOSIStemp`
if ( $#return != 0 ) then
  echo "Load of database dump $DatabaseDumps failed"
  echo "     Please correct below errors and rerun"
  cat DOSIStemp
  exit(1)
endif
echo "DOSIS database dump loaded from $DatabaseDumps"
#
# load transaction dumps
#
set LogDumps = ( $DOSIS_COMMAND_DIR/check*_vars $DOSIS_BACKUP/*.tld )
shift LogDumps
while ( $#LogDumps > 0 )
  sort <<EXIT >DOSIStemp
$LogDumps[1]:r
$DatabaseDumps:r
EXIT
  set return = `cat DOSIStemp`
  if ( $return[1] == $DatabaseDumps:r && $return[2] != $DatabaseDumps:r ) then
    cp $LogDumps[1] $DOSIS_DUMP_FILE
    if ( $status != 0 ) then
      echo "Copy of transaction log dump $LogDumps[1] failed"
      echo "     Please correct and rerun"
      exit(1)
    endif
    isql -Usa -P$SA_PASSWORD <<EXIT >DOSIStemp
load transaction DOSIS from $DOSIS_DUMP_DEVICE
go
EXIT
    set return = `grep msg DOSIStemp`
    if ( $#return != 0 ) then
      echo "Load of transaction log $LogDumps[1] failed"
      echo "     Please correct below errors and rerun"
      cat DOSIStemp
      exit(1)
    endif
    echo "DOSIS transaction log loaded from $LogDumps[1]"
  endif
  shift LogDumps
end
#
# success message
#
echo "DOSIS restore complete"
exit(0)
```

COPY ARCHIVED FILES FUNCTION SCRIPTING

```
#!csh
#
# dosis_tape [ yyyymmdd ] [ DEV ]
#   copy archive files for yyyymmdd (if specified), else oldest files
#   to /dev/DEV (if specified), else to /dev/$DOSIS_TAPE
#
unset noclobber
#
# make sure all required environment variables are set correctly
#
if ( $?DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR not set; please set and rerun"
  set DOSIS_COMMAND_DIR = dummy
  exit(1)
else if ( -d $DOSIS_COMMAND_DIR == 0 ) then
  echo "Environment variable DOSIS_COMMAND_DIR does not designate a directory"
  echo "      Please set and rerun"
  exit(1)
endif
if ( -e $DOSIS_COMMAND_DIR/check_vars == 0 ) then
  echo "$DOSIS_COMMAND_DIR/check_vars script does not exist"
  echo "      Please fix and rerun"
  exit(1)
endif
source $DOSIS_COMMAND_DIR/check_vars
if ( $status != 0 ) exit(1)
#
# check on how many arguments we have; set Date and Tape as appropriate
#
unset Date
unset Tape
switch ( $#argv )
case 0:
  breaksw
case 1:
  egrep 'marker19|marker20' <<EXIT >DOSIStemp
marker$argv[1]
EXIT
  set return = `cat DOSIStemp`
  if ( $#return == 1 ) then
    set Date = $argv[1]
  else
    set Tape = $argv[1]
  endif
  breaksw
case 2:
default:
  set Date = $argv[1]
  set Tape = $argv[2]
  breaksw
endsw
#
# find oldest archived files if date is not specified
#
if ( $?Date == 0 ) then
  set Date = ( $DOSIS_COMMAND_DIR/check*_vars $DOSIS_ARCHIVE/*.ad )
  shift Date
  if ( $#Date == 0 ) then
    echo "No archive files are present.  Please take an archive and rerun"
    exit(1)
  endif
```

```
 awk '{ print substr($1,1,8) }' <<EXIT >DOSIStemp
$Date[1]:t
EXIT
 set Date = `cat DOSIStemp`
endif
#
# set Tape if tape has not been specified
#
if ( $?Tape == 0 ) then
  if ( $?DOSIS_TAPE == 0 ) then
    echo "You must specify a tape drive, or set DOSIS_TAPE environment var."
    echo "      Please correct and rerun"
    exit(1)
  endif
  set Tape = $DOSIS_TAPE
endif
set Tape = /dev/$Tape
#
# make sure Tape device is valid
#
if ( -e $Tape == 0 ) then
  echo "Tape device $Tape does not exist.  Please check and rerun"
  exit(1)
endif
#
# make sure all archive files exist
#
set WorkArchiveTables = ( $ArchiveTables )
unset error
while ( $#WorkArchiveTables != 0 )
  set filename = $DOSIS_ARCHIVE/$Date$WorkArchiveTables[2].ad
  if ( -e $filename == 0 ) then
    echo "Archive file $filename does not exist.  Please correct and rerun"
    set error = 1
  endif
  shift WorkArchiveTables
  shift WorkArchiveTables
  shift WorkArchiveTables
end
if ( $?error ) then
  echo "Please correct above errors and rerun"
  exit(1)
endif
#
# make the tape
#
( cd $DOSIS_ARCHIVE ; tar cf $Tape $Date*.ad )
if ( $status != 0 ) then
  echo "Creation of archive tape failed: Please correct above errors and rerun"
  exit(1)
endif
echo "$Date archive written to $Tape"
exit(0)
```

Appendix C
Configuration Management Library Structure

The format of the file listings within the Configuration Management Library Structure is as follows:

/directory-name/directory-name/.../file-name

The files contained within the Silent Inspector system are:

```
/BINDING/vadscrt1.0
/BINDING/libasybdb.a
/BINDING/libsybdb.a
/BINDING/v_usr_conf.a
/BINDING/v_usr_conf_b.a
/BINDING/v_usr_conf_i.a
/BINDING/c_timeB.o
/BINDING/xev.o
/SAME/basetypeS.a
/SAME/booleanS.a
/SAME/charB.a
/SAME/charS.a
/SAME/communicB.a
/SAME/communicS.a
/SAME/db_errorB.a
/SAME/dateS.a
/SAME/db_errorS.a
/SAME/doubleB.a
/SAME/doubleS.a
/SAME/enumB.a
/SAME/enumS.a
/SAME/intB.a
/SAME/exceptS.a
/SAME/intS.a
/SAME/realB.a
/SAME/realS.a
/SAME/to_strinX.a
/SAME/smallintB.a
/SAME/smallintS.a
/SAME/not_nullX.a
/SAME/sourceS.a
/SAME/standardS.a
/SAME/systemS.a
/SAME/booleanB.a
/SL-MODELS/s.proj_add.g
/SL-MODELS/p.Makefile
/SL-MODELS/s.Makefile
/SL-MODELS/s.mk_button.g
/SL-MODELS/s.mfiledbl.g
/SL-MODELS/s.fldent_sz.g
/SL-MODELS/Makefile
/SL-MODELS/s.m_return.g
/SL-MODELS/s.fldent85.g
/SL-MODELS/s.m_call1.g
/SL-MODELS/s.timetrend.g
/SL-MODELS/CorpLogo.m1
/SL-MODELS/s.N_call.g
/SL-MODELS/s.da_twobut.g
/SL-MODELS/s.m_1_column.g
/SL-MODELS/M_call.m1
/SL-MODELS/M_default.m1
/SL-MODELS/s.m_quit.g
/SL-MODELS/s.daily.g
/SL-MODELS/s.day_foot.g
/SL-MODELS/s.day_head.g
/SL-MODELS/s.day_line.g
/SL-MODELS/s.day_total.g
/SL-MODELS/s.day_totft.g
/SL-MODELS/s.day_tothd.g
/SL-MODELS/s.m_call.g
/SL-MODELS/N_call.m1
/SL-MODELS/Informatio.m1
```

/SL-MODELS/Warning.m1
/SL-MODELS/s.floatout.g
/SL-MODELS/background.m1
/SL-MODELS/backdrop.m1
/SL-MODELS/button.m1
/SL-MODELS/choice.m1
/SL-MODELS/control.m1
/SL-MODELS/da_twobut.m1
/SL-MODELS/s.legend.g
/SL-MODELS/daily.m1
/SL-MODELS/s.mo_pushbut.g
/SL-MODELS/s.m_dialog.g
/SL-MODELS/s.m_dnarrow.g
/SL-MODELS/s.m_slidehx.g
/SL-MODELS/s.m_slider.g
/SL-MODELS/day_foot.m1
/SL-MODELS/s.m_slidehxl.g
/SL-MODELS/s.m_slidevxl.g
/SL-MODELS/s.m_txtscrol.g
/SL-MODELS/s.m_uparrow.g
/SL-MODELS/s.mfilesel.g
/SL-MODELS/s.md_button2.g
/SL-MODELS/s.pushface.g
/SL-MODELS/day_head.m1
/SL-MODELS/s.plot.g
/SL-MODELS/s.panel.g
/SL-MODELS/day_line.m1
/SL-MODELS/day_total.m1
/SL-MODELS/day_totft.m1
/SL-MODELS/day_tothd.m1
/SL-MODELS/do_twobut.m1
/SL-MODELS/downtime.m1
/SL-MODELS/s.stamp.g
/SL-MODELS/s.plt_twobut.g
/SL-MODELS/s.scrollst.g
/SL-MODELS/s.CorpLogo.g
/SL-MODELS/s.Informatio.g
/SL-MODELS/s.M_call.g
/SL-MODELS/s.M_default.g
/SL-MODELS/s.Warning.g
/SL-MODELS/s.background.g
/SL-MODELS/s.quitbutton.g
/SL-MODELS/s.button.g
/SL-MODELS/s.choice.g
/SL-MODELS/s.control.g
/SL-MODELS/s.reportbar.g
/SL-MODELS/s.reportpg.g
/SL-MODELS/s.reportst.g
/SL-MODELS/dr_ops.m1
/SL-MODELS/s.scrollbox.g
/SL-MODELS/draftx.m1
/SL-MODELS/s.text_sz.g
/SL-MODELS/s.text_fz.g
/SL-MODELS/s.text_left.g
/SL-MODELS/s.textout.g
/SL-MODELS/s.do_twobut.g
/SL-MODELS/s.downtime.g
/SL-MODELS/s.dr_ops.g
/SL-MODELS/s.draftx.g
/SL-MODELS/s.ds_pushbut.g
/SL-MODELS/s.dss.g
/SL-MODELS/s.dssentry.g
/SL-MODELS/s.dssscreen.g
/SL-MODELS/s.dt_input.g
/SL-MODELS/s.entry.g
/SL-MODELS/ds_pushbut.m1
/SL-MODELS/s.gen_head.g
/SL-MODELS/s.intro.g
/SL-MODELS/s.job.g
/SL-MODELS/s.job_head.g
/SL-MODELS/dss.m1
/SL-MODELS/s.top.g
/SL-MODELS/s.text_box.g
/SL-MODELS/dssentry.m1
/SL-MODELS/dssscreen.m1
/SL-MODELS/dt_input.m1
/SL-MODELS/entry.m1
/SL-MODELS/fldent85.m1
/SL-MODELS/floatout.m1
/SL-MODELS/intro.m1
/SL-MODELS/legend.m1
/SL-MODELS/s.md_button.g
/SL-MODELS/m_1_column.m1
/SL-MODELS/m_call.m1
/SL-MODELS/m_call1.m1
/SL-MODELS/s.motifscale.g
/SL-MODELS/s.noptions.g
/SL-MODELS/s.options.g
/SL-MODELS/s.outputdbl.g
/SL-MODELS/s.outputonly.g
/SL-MODELS/s.outputwide.g
/SL-MODELS/m_dialog.m1
/SL-MODELS/m_dnarrow.m1
/SL-MODELS/s.trip_line.g
/SL-MODELS/s.print.g
/SL-MODELS/s.pushbutton.g
/SL-MODELS/s.trip_total.g
/SL-MODELS/m_quit.m1
/SL-MODELS/m_return.m1
/SL-MODELS/m_slideh.m1
/SL-MODELS/m_slidehx.m1
/SL-MODELS/m_slidehxl.m1
/SL-MODELS/m_slider.m1
/SL-MODELS/s.select4.g
/SL-MODELS/m_slidevxl.m1
/SL-MODELS/m_txtscrol.m1
/SL-MODELS/m_uparrow.m1
/SL-MODELS/md_button2.m1
/SL-MODELS/md_button.m1
/SL-MODELS/mfilesel.m1
/SL-MODELS/mk_button.m1
/SL-MODELS/mo_pushbut.m1
/SL-MODELS/s.trip.g
/SL-MODELS/s.trip_foot.g
/SL-MODELS/s.trip_head.g
/SL-MODELS/motifscale.m1
/SL-MODELS/navaid.m1
/SL-MODELS/noptions.m1
/SL-MODELS/options.m1
/SL-MODELS/outputdbl.m1
/SL-MODELS/outputonly.m1
/SL-MODELS/outputwide.m1
/SL-MODELS/panel.m1
/SL-MODELS/plot.m1

```
/SL-MODELS/plt_twobut.m1
/SL-MODELS/portland.m1
/SL-MODELS/print.m1
/SL-MODELS/pushbutton.m1
/SL-MODELS/pushface.m1
/SL-MODELS/quitbutton.m1
/SL-MODELS/reportbar.m1
/SL-MODELS/reportpg.m1
/SL-MODELS/reportst.m1
/SL-MODELS/s.m_slideh.g
/SL-MODELS/s.backdrop.g
/SL-MODELS/scrollbox.m1
/SL-MODELS/scrollst.m1
/SL-MODELS/showmap.m1
/SL-MODELS/s.navaid.g
/SL-MODELS/s.portland.g
/SL-MODELS/stamp.m1
/SL-MODELS/c.awk
/SL-MODELS/text_box.m1
/SL-MODELS/text_fz.m1
/SL-MODELS/text_left.m1
/SL-MODELS/text_sz.m1
/SL-MODELS/textout.m1
/SL-MODELS/timetrend.m1
/SL-MODELS/top.m1
/SL-MODELS/trip.m1
/SL-MODELS/s.showmap.g
/SL-MODELS/s.value.g
/SL-MODELS/trip_foot.m1
/SL-MODELS/trip_head.m1
/SL-MODELS/trip_line.m1
/SL-MODELS/trip_total.m1
/SL-MODELS/value.m1
/SL-MODELS/job.m1
/SL-MODELS/job_head.m1
/SL-MODELS/select4.m1
/SYBASE/.imports/STANDARD
/SYBASE/.imports/a_strings
/SYBASE/.imports/file_support
/SYBASE/.imports/io_exceptions
/SYBASE/.imports/number_io
/SYBASE/.imports/os_files
/SYBASE/.imports/system
/SYBASE/.imports/text_io
/SYBASE/.imports/text_io.intege
/SYBASE/.imports/text_ioB
/SYBASE/.imports/text_supprt
/SYBASE/.imports/text_supprtB
/SYBASE/.imports/unchecked_conv
/SYBASE/.imports/unsigned_types
/SYBASE/.lines/INST13XX
/SYBASE/.lines/INST14XX
/SYBASE/.lines/.bara02
/SYBASE/.lines/.bcpb01
/SYBASE/.lines/.err_hd_ctrb01
/SYBASE/.lines/.err_hd_ctrs01
/SYBASE/.lines/.example2pca01
/SYBASE/.lines/.handlersb01
/SYBASE/.lines/.ifaceb01
/SYBASE/.lines/.msg_hd_ctrb01
/SYBASE/.lines/.msg_hd_ctrs01
/SYBASE/.lines/.proc_evntsb01
/SYBASE/.lines/.str_utilb01
/SYBASE/.lines/.two_phb01
/SYBASE/.nets/INST13XX
/SYBASE/.nets/INST14XX
/SYBASE/.nets/.Verdixs01
/SYBASE/.nets/.Verdixs02
/SYBASE/.nets/.bara01
/SYBASE/.nets/.bara02
/SYBASE/.nets/.bcp_lows01
/SYBASE/.nets/.bcp_lows02
/SYBASE/.nets/.bcpb01
/SYBASE/.nets/.bcps01
/SYBASE/.nets/.c_err_hds01
/SYBASE/.nets/.c_err_hds02
/SYBASE/.nets/.c_msg_hds01
/SYBASE/.nets/.c_msg_hds02
/SYBASE/.nets/.err_hd_ctrb01
/SYBASE/.nets/.err_hd_ctrs01
/SYBASE/.nets/.err_hdb01
/SYBASE/.nets/.err_hds01
/SYBASE/.nets/.example2pca01
/SYBASE/.nets/.handlersb01
/SYBASE/.nets/.handlerss01
/SYBASE/.nets/.iface_lows01
/SYBASE/.nets/.iface_lows02
/SYBASE/.nets/.ifaceb01
/SYBASE/.nets/.ifaces01
/SYBASE/.nets/.msg_hd_ctrb01
/SYBASE/.nets/.msg_hd_ctrs01
/SYBASE/.nets/.msg_hdb01
/SYBASE/.nets/.msg_hds01
/SYBASE/.nets/.proc_evntsb01
/SYBASE/.nets/.proc_evntss01
/SYBASE/.nets/.scounixs01
/SYBASE/.nets/.scounixs02
/SYBASE/.nets/.str_utilb01
/SYBASE/.nets/.str_utils01
/SYBASE/.nets/.two_ph_lows01
/SYBASE/.nets/.two_ph_lows02
/SYBASE/.nets/.two_phb01
/SYBASE/.nets/.two_phs01
/SYBASE/.nets/.typess01
/SYBASE/.nets/.typess02
/SYBASE/.objects/INST13XX
/SYBASE/.objects/INST14XX
/SYBASE/.objects/example1.o
/SYBASE/.objects/example2.o
/SYBASE/.objects/example3.o
/SYBASE/.objects/example4.o
/SYBASE/.objects/example5.o
/SYBASE/.objects/example6.o
/SYBASE/.objects/example7.o
/SYBASE/.objects/example8.o
/SYBASE/.objects/.Verdixs01
/SYBASE/.objects/.Verdixs02
/SYBASE/.objects/.bara01
/SYBASE/.objects/.bara02
/SYBASE/.objects/.bcp_lows01
/SYBASE/.objects/.bcp_lows02
/SYBASE/.objects/.bcpb01
/SYBASE/.objects/.bcps01
/SYBASE/.objects/.c_err_hds01
/SYBASE/.objects/.c_err_hds02
/SYBASE/.objects/.c_msg_hds01
```

```
/SYBASE/.objects/.c_msg_hds02
/SYBASE/.objects/.err_hd_ctrb01
/SYBASE/.objects/.err_hd_ctrs01
/SYBASE/.objects/.example2pca01
/SYBASE/.objects/.handlersb01
/SYBASE/.objects/.handlerss01
/SYBASE/.objects/.iface_lows01
/SYBASE/.objects/.iface_lows02
/SYBASE/.objects/.ifaceb01
/SYBASE/.objects/.ifaces01
/SYBASE/.objects/.msg_hd_ctrb01
/SYBASE/.objects/.msg_hd_ctrs01
/SYBASE/.objects/.proc_evntsb01
/SYBASE/.objects/.proc_evntss01
/SYBASE/.objects/.scounixs01
/SYBASE/.objects/.scounixs02
/SYBASE/.objects/.str_utilb01
/SYBASE/.objects/.str_utils01
/SYBASE/.objects/.two_ph_lows01
/SYBASE/.objects/.two_ph_lows02
/SYBASE/.objects/.two_phb01
/SYBASE/.objects/.two_phs01
/SYBASE/.objects/.typess01
/SYBASE/.objects/.typess02
/database/Makefile
/database/adm_conS.a
/database/adminS.a
/database/db_stdS.a
/database/disp_conS.a
/database/disposalS.a
/database/down_conS.a
/database/downtimeS.a
/database/dr_conS.a
/database/dredgeS.a
/database/enum_indS.a
/database/flt_indS.a
/database/gsqlutlS.a
/database/gsybutlS.a
/database/int_indS.a
/database/lddisconS.a
/database/ldstaconS.a
/database/loadS.a
/database/load_conS.a
/database/loaddispS.a
/database/loadstatS.a
/database/prdisconS.a
/database/proj_conS.a
/database/projdispS.a
/database/projdredS.a
/database/projectS.a
/database/projstatS.a
/database/projsumS.a
/database/prstaconS.a
/database/prsumconS.a
/database/sta_conS.a
/database/stateS.a
/database/stateconS.a
/database/stationS.a
/database/str_indS.a
/database/syb_lowS.a
/database/syb_utlS.a
/database/sql_utlS.a
/database/prdr_conS.a
/database/adm_conB.a
/database/adminB.a
/database/databaseB.a
/database/disp_conB.a
/database/disposalB.a
/database/down_conB.a
/database/downtimeB.a
/database/dr_conB.a
/database/dredgeB.a
/database/enum_indB.a
/database/flt_indB.a
/database/gsqlutlB.a
/database/gsybutlB.a
/database/int_indB.a
/database/lddisconB.a
/database/ldstaconB.a
/database/loadB.a
/database/load_conB.a
/database/loaddispB.a
/database/loadstatB.a
/database/prdisconB.a
/database/prdr_conB.a
/database/proj_conB.a
/database/projdispB.a
/database/projdredB.a
/database/projectB.a
/database/projstatB.a
/database/projsumB.a
/database/prstaconB.a
/database/prsumconB.a
/database/sql_utlB.a
/database/sta_conB.a
/database/stateB.a
/database/stateconB.a
/database/stationB.a
/database/str_indB.a
/database/syb_utlB.a
/dss/RANGE.DAT
/dss/Makefile
/dss/CorpLogo.m1
/dss/stamp.m1
/dss/button.m1
/dss/pushface.m1
/dss/mfilesel.m1
/dss/motifscale.m1
/dss/dssentry.m1
/dss/quitbutton.m1
/dss/panel.m1
/dss/outputwide.m1
/dss/print.m1
/dss/outputdbl.m1
/dss/outputonly.m1
/dss/background.m1
/dss/ds_pushbut.m1
/dss/pushbutton.m1
/dss/options.m1
/dss/noptions.m1
/dss/dssscreen.m1
/dss/choice.m1
/dss/intro.m1
/dss/entry.m1
```

```
/dss/dss.m1
/dss/control.m1
/dss/controlS.a
/dss/entryS.a
/dss/gismosS.a
/dss/input_fiS.a
/dss/sensorS.a
/dss/sinitS.a
/dss/dssA.a
/dss/sinitB.a
/dss/controlB.a
/dss/entryB.a
/dss/gismosB.a
/dss/input_fiB.a
/dss/sensorB.a
/ship/monitor/Makefile
/ship/monitor/timetrend.m1
/ship/monitor/stamp.m1
/ship/monitor/textout.m1
/ship/monitor/text_sz.m1
/ship/monitor/pushface.m1
/ship/monitor/mo_pushbut.m1
/ship/monitor/panel.m1
/ship/monitor/legend.m1
/ship/monitor/floatout.m1
/ship/monitor/draftx.m1
/ship/monitor/background.m1
/ship/monitor/dr_ops.m1
/ship/monitor/dr_opsS.a
/ship/monitor/sinitS.a
/ship/monitor/dr_opsB.a
/ship/monitor/sinitB.a
/ship/monitor/monitorA.a
/ship/reports/trip/Makefile
/ship/reports/trip/Informatio.m1
/ship/reports/trip/background.m1
/ship/reports/trip/m_dialog.m1
/ship/reports/trip/pushface.m1
/ship/reports/trip/quitbutton.m1
/ship/reports/trip/mfilesel.m1
/ship/reports/trip/text_sz.m1
/ship/reports/trip/text_fz.m1
/ship/reports/trip/text_left.m1
/ship/reports/trip/stamp.m1
/ship/reports/trip/trip.m1
/ship/reports/trip/trip_foot.m1
/ship/reports/trip/trip_head.m1
/ship/reports/trip/trip_line.m1
/ship/reports/trip/trip_total.m1
/ship/reports/trip/M_call.m1
/ship/reports/trip/M_default.m1
/ship/reports/trip/N_call.m1
/ship/reports/trip/Warning.m1
/ship/reports/trip/m_1_column.m1
/ship/reports/trip/m_call.m1
/ship/reports/trip/m_dnarrow.m1
/ship/reports/trip/m_slidehx.m1
/ship/reports/trip/m_slidehxl.m1
/ship/reports/trip/m_slider.m1
/ship/reports/trip/m_slidevxl.m1
/ship/reports/trip/m_txtscrol.m1
/ship/reports/trip/m_uparrow.m1
/ship/reports/trip/scrollbox.m1
/ship/reports/trip/top.m1
/ship/reports/trip/reportbar.m1
/ship/reports/trip/reportpg.m1
/ship/reports/trip/reportst.m1
/ship/reports/trip/scrollst.m1
/ship/reports/trip/tripS.a
/ship/reports/trip/tripA.a
/ship/reports/trip/autotripA.a
/ship/reports/trip/tripB.a
/ship/reports/trip/autotrip
/ship/reports/daily/Makefile
/ship/reports/daily/M_call.m1
/ship/reports/daily/M_default.m1
/ship/reports/daily/N_call.m1
/ship/reports/daily/Warning.m1
/ship/reports/daily/background.m1
/ship/reports/daily/daily.m1
/ship/reports/daily/day_foot.m1
/ship/reports/daily/day_head.m1
/ship/reports/daily/day_line.m1
/ship/reports/daily/day_total.m1
/ship/reports/daily/day_totft.m1
/ship/reports/daily/day_tothd.m1
/ship/reports/daily/m_1_column.m1
/ship/reports/daily/m_call.m1
/ship/reports/daily/m_dialog.m1
/ship/reports/daily/m_dnarrow.m1
/ship/reports/daily/m_slidehx.m1
/ship/reports/daily/m_slidehxl.m1
/ship/reports/daily/m_slider.m1
/ship/reports/daily/m_slidevxl.m1
/ship/reports/daily/m_txtscrol.m1
/ship/reports/daily/m_uparrow.m1
/ship/reports/daily/mfilesel.m1
/ship/reports/daily/pushface.m1
/ship/reports/daily/quitbutton.m1
/ship/reports/daily/reportbar.m1
/ship/reports/daily/reportpg.m1
/ship/reports/daily/reportst.m1
/ship/reports/daily/scrollbox.m1
/ship/reports/daily/scrollst.m1
/ship/reports/daily/stamp.m1
/ship/reports/daily/text_fz.m1
/ship/reports/daily/text_left.m1
/ship/reports/daily/text_sz.m1
/ship/reports/daily/top.m1
/ship/reports/daily/da_twobut.m1
/ship/reports/daily/dailyS.a
/ship/reports/daily/dailyA.a
/ship/reports/daily/autodalyA.a
/ship/reports/daily/dailyB.a
/ship/reports/job/Makefile
/ship/reports/job/job.m1
/ship/reports/job/job_head.m1
/ship/reports/job/Informatio.m1
/ship/reports/job/background.m1
/ship/reports/job/do_twobut.m1
/ship/reports/job/m_dialog.m1
/ship/reports/job/pushface.m1
/ship/reports/job/quitbutton.m1
/ship/reports/job/mfilesel.m1
/ship/reports/job/text_sz.m1
/ship/reports/job/text_fz.m1
```

```
/ship/reports/job/text_left.m1
/ship/reports/job/stamp.m1
/ship/reports/job/M_call.m1
/ship/reports/job/M_default.m1
/ship/reports/job/N_call.m1
/ship/reports/job/Warning.m1
/ship/reports/job/m_1_column.m1
/ship/reports/job/m_call.m1
/ship/reports/job/m_dnarrow.m1
/ship/reports/job/m_slidehx.m1
/ship/reports/job/m_slidehxl.m1
/ship/reports/job/m_slider.m1
/ship/reports/job/m_slidevxl.m1
/ship/reports/job/m_txtscrol.m1
/ship/reports/job/m_uparrow.m1
/ship/reports/job/md_button.m1
/ship/reports/job/scrollbox.m1
/ship/reports/job/top.m1
/ship/reports/job/reportbar.m1
/ship/reports/job/reportpg.m1
/ship/reports/job/reportst.m1
/ship/reports/job/scrollst.m1
/ship/reports/job/jobS.a
/ship/reports/job/jobA.a
/ship/reports/job/jobB.a
/ship/downtime/Makefile
/ship/downtime/N_call.m1
/ship/downtime/background.m1
/ship/downtime/downtime.m1
/ship/downtime/dt_input.m1
/ship/downtime/m_call.m1
/ship/downtime/md_button.m1
/ship/downtime/fldent85.m1
/ship/downtime/mfilesel.m1
/ship/downtime/pushface.m1
/ship/downtime/stamp.m1
/ship/downtime/scrollst.m1
/ship/downtime/scrollbox.m1
/ship/downtime/m_txtscrol.m1
/ship/downtime/m_dnarrow.m1
/ship/downtime/m_uparrow.m1
/ship/downtime/m_slider.m1
/ship/downtime/m_1_column.m1
/ship/downtime/mk_button.m1
/ship/downtime/text_fz.m1
/ship/downtime/text_sz.m1
/ship/downtime/do_twobut.m1
/ship/downtime/dfldprocS.a
/ship/downtime/dtinputS.a
/ship/downtime/dtselectS.a
/ship/downtime/downtimeA.a
/ship/downtime/dfldprocB.a
/ship/downtime/dtinputB.a
/ship/downtime/dtselectB.a
/ship/plot/Makefile
/ship/plot/plotS.a
/ship/plot/plotA.a
/ship/plot/plotB.a
/ship/rtkernelA.a
/support/Makefile
/support/cstringsS.a
/support/coorS.a
/support/debugA.a
/support/dialogstS.a
/support/domainS.a
/support/fieldmgrS.a
/support/genericstS.a
/support/getenvA.a
/support/graphstS.a
/support/id_serverS.a
/support/mapS.a
/support/menubarstS.a
/support/randomS.a
/support/reportstS.a
/support/scrollstS.a
/support/searchS.a
/support/supportS.a
/support/termioS.a
/support/timeS.a
/support/to_sA.a
/support/to_vA.a
/support/todS.a
/support/vadsinitS.a
/support/databaseS.a
/support/graphS.a
/support/rep_genS.a
/support/dredgeioS.a
/support/computeS.a
/support/computeB.a
/support/coorB.a
/support/cstringsB.a
/support/c127strS.a
/support/dialogstB.a
/support/dredgeioB.a
/support/fieldmgrB.a
/support/genericstB.a
/support/graphB.a
/support/graphstB.a
/support/id_serverB.a
/support/mapB.a
/support/menubarstB.a
/support/randomB.a
/support/rep_genB.a
/support/reportstB.a
/support/scrollstB.a
/support/searchB.a
/support/supportB.a
/support/timeB.a
/support/todB.a
/support/vadsinitB.a
```

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC20503.

1. **AGENCY USE ONLY *(Leave blank)***	2. **REPORT DATE** February 1996	3. **REPORT TYPE AND DATES COVERED** Final report

4. **TITLE AND SUBTITLE** Silent Inspector System Technical Manual	5. **FUNDING NUMBERS**
6. **AUTHOR(S)** Jeffrey M. Cox, Paul Maresca, James Rosati III	
7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Evans-Hamilton, Inc. 731 Northlake Way, Suite 201, Seattle, WA 98103 AdaSoft, Inc. 8750-9 Cherry Lane, Laurel, MD 20707 U.S. Army Engineer Waterways Experiment Station 3909 Halls Ferry Road, Vicksburg, MS 39180-6199	8. **PERFORMING ORGANIZATION REPORT NUMBER** Technical Report DRP-96-1
9. **SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** U.S. Army Corps of Engineers Washington, DC 20314-1000	10. **SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. **SUPPLEMENTARY NOTES**

Available from National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

12a. **DISTRIBUTION/AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited.	12b. **DISTRIBUTION CODE**

13. **ABSTRACT *(Maximum 200 words)***

This report describes the Silent Inspector system developed for monitoring hopper dredge operations. The system collects and records measurements from shipboard sensors, calculates the dredging activities being performed and the weight of the material being recovered, and displays this information through standard reports and graphical data displays. Recorded data are also automatically backed up and later archived to allow transfer of the data to other locations. The Silent Inspector data can provide a permanent record of the dredging activity.

This report is intended to be used by systems engineers. A companion report entitled *Silent Inspector User's Manual* (Cox, Maresca, and Jarvela 1995) published by the U.S. Army Engineer Waterways Experiment Station describes how to operate the installed system.

14. **SUBJECT TERMS** Dredging Hopper dredges Silent Inspector System	15. **NUMBER OF PAGES** 126
	16. **PRICE CODE**

17. **SECURITY CLASSIFICATION OF REPORT** UNCLASSIFIED	18. **SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	19. **SECURITY CLASSIFICATION OF ABSTRACT**	20. **LIMITATION OF ABSTRACT**

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Destroy this report when no longer needed. Do not return it to the originator.